Vers une approche de découverte et de sélection distribuées des services IoT basées sur des avatars autonomes

Karima Khadir

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *22/02/2021* **par :**

KARIMA KHADIR

Vers une approche de découverte et de sélection distribuées des services IoT basées sur des avatars autonomes

---

JURY

CHIHAB HANACHI	Professeur d'Université	Président du jury
MICHAEL MRISSA	Professeur d'Université	Rapporteur
BOUALEM BENATALLAH	Professeur d'Université	Rapporteur
CHIRINE CHEDIRA GUEGAN	Professeur d'Université	Examinatrice
THIERRY MONTEIL	Professeur d'Université	Co-directeur de thèse
NAWAL GUERMOUCHE	Maître de Conférences	Co-directrice de thèse

---

École doctorale et spécialité :

MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :

Laboratoire d'analyse et d'architecture des systèmes

Directeur(s) de Thèse :

Thierry MONTEIL et *Nawal GUERMOUCHE*

Rapporteurs :

Michael Mrissa et *Boualem Benatallah*

Je dédie ce modeste travail,
A mon cher époux Arezki. Depuis le jour où je t'ai connu, ma vie est comblée de bonheur.
A ma chère maman qui s'est beaucoup sacrifiée pour moi et qui a été toujours là pour moi,
A mon père, qui m'a donné le plus grand courage et qui s'est toujours sacrifié pour me voir réussir,
A mes frères, Smail et Hamid,
A tous mes amis et proches.

Remerciements

Arrivant au terme de ce travail, je tiens à exprimer dans ces lignes ma reconnaissance la plus sincère à toutes les personnes qui ont participé d'une manière ou d'une autre à son aboutissement.

En premier lieu, j'adresse mes plus cordiaux remerciements à Mme. Nawal GUERMOUCHE et M. Thierry MONTEIL, mes directeurs de thèse, pour leur disponibilité et leurs précieux conseils. Ils n'ont jamais manqué de me conseiller et de m'orienter tout au long de mon travail. Qu'ils trouvent ici l'expression de mon respect et de ma profonde gratitude.

Un grand merci à M. Michael Mrissa et M. Boualem Benatallah d'avoir accepté de rapporter mon mémoire. Je remercie également M. Chihab Hanachi et Mme. Chirine Chedira Guegan d'être mes examinateurs. Je tiens à les remercier tous pour avoir accepté de participer à mon jury de thèse et M. Chihab Hanachi pour l'honneur qu'il me fait de présider ce jury. Je remercie également M. Olivier Guerard, de la part de Continental Automotive, pour sa participation scientifique ainsi que le temps qu'il a consacré à ma recherche.

Un grand merci à tous les membres de l'équipe SARA pour leur chaleureux accueil et tout ce qui ont contribué à ce que cette thèse soit réalisée dans une bonne ambiance. Je remercie particulièrement mes collègues de bureau Guillaume, François, Nicolas, Laurent et Santi pour les moments agréables qu'on a passé ensemble. Je n'oublie pas de remercier chaleureusement les secrétaires de l'équipe : Justine Praneuf, Belle Urica Rakotomalala et Marie-Agnès Bellieres pour leur professionnalisme, leur efficacité ainsi que leur bonne humeur. Je remercie également le personnel des enseignants et des administratifs de l'INSA pour leur aide pendant mes interventions en tant que vacataire et DCE.

Je voudrais exprimer ma reconnaissance envers mes amis et mes collèges Baya, Yasmine, Souad, Walid, Tanissia, Nour, Raoua, Soufian qui m'ont apporté leur soutien moral et intellectuel tout au long de ces trois ans passés.

Tous mes remerciements et ma gratitude à mon cher époux qui était toujours là pour moi et qui n'a jamais cessé de me soutenir, de m'encourager et me redonner confiance en moi dans mes plus difficiles moments de ce projet scientifique.

Je ne pourrais pas clôturer ces quelques lignes sans témoigner mes plus vifs remerciements à ma famille : ma mère, mon père, ma belle-mère, mon beau-père, mes frères Hamid et Smail pour leur soutien continu et leurs encouragements.

À ma grande famille, celle du sang ou de choix : MERCI.

Résumé et Mots Clés

Résumé

L'Internet des Objets (Internet of Things, IoT) est défini comme une infrastructure complexe composée d'un grand nombre de dispositifs hétérogènes. Il s'appuie sur les technologies de l'information et de la communication pour interconnecter ces appareils et pour leur permettre d'échanger leurs données pour fournir une variété de services à valeur ajoutée. Le Web des Objets (Web of Things, WoT) permet de mettre en œuvre cette vision en s'appuyant sur le Web pour virtualiser et interconnecter les objets IoT de manière transparente. Les systèmes IoT fonctionnent souvent dans des environnements dynamiques, imprévisibles et mobiles avec des ressources limitées. Par conséquent, les objets IoT virtualisés sur le Web ont besoin de capacités de raisonnement et de prise de décision autonomes pour adapter leurs comportements à leur contexte. Pour cela, nous proposons d'étendre les objets virtuels passifs avec des mécanismes de raisonnement basés sur des modèles sémantiques. Cette extension est appelée *avatar autonome*.

Un avatar peut fournir plusieurs services qui peuvent être découverts, récupérés et composés pour créer de nouvelles applications IoT grâce à des techniques de découverte et de sélection de services. Cependant, avec la prolifération et l'évolution exceptionnelle du nombre d'objets connectés, ces méthodes deviennent problématiques. Pour surmonter ces défis, nous proposons l'intégration des concepts des réseaux sociaux pour l'IoT. Ils permettent de faire émerger l'intelligence collective d'un groupe d'avatars qui peuvent fournir des réponses beaucoup plus intéressantes et riches à des problèmes complexes qu'un seul avatar. De plus, ils accélèrent et facilitent la navigabilité dans un réseau IoT dynamique à large échelle. Pour intégrer ce concept, nous proposons une approche s'appuyant sur la notion de distance sociale. Pour réduire l'espace de recherche de découverte et mieux cibler la transmission des requêtes aux avatars appropriés, une méthode de clustering basée sur l'algorithme fuzzy c-means est proposée pour classer les avatars sociaux selon leurs fonctionnalités.

Les services IoT découverts possèdent des caractéristiques non-fonctionnelles différentes notamment en termes de qualité de service (QoS). Dans ce contexte, nous proposons une nouvelle approche de sélection distribuée qui tient compte des paramètres de QoS ainsi que de leur fluctuation. Une première approche basée sur un algorithme génétique évolutionnaire multi-objectifs (MOEA, Multi- Objective Evolutionary Algorithm) est proposée afin de décomposer les contraintes de QoS globales en un ensemble de contraintes locales permettant ainsi d'effectuer une sélection locale qui garantit le respect des contraintes globales. Cette approche a été ensuite étendue par des mécanismes d'apprentissage automatique supervisé pour prédire les contraintes locales. Pour ce faire, nous avons proposé une approche basée sur SVM (Support Vector Machine).

Les approches proposées sont évaluées dans le contexte des véhicules connectés et via une série d'expérimentations. Les résultats expérimentaux montrent l'efficacité de nos approches pour la découverte et la sélection de services et les avantages qu'elles fournissent notamment par rapport aux approches traditionnelles centralisées.

Mots Clés : IoT, Avatar, Sémantique, Réseaux sociaux, Clustering, QoS, MOEA, SVM, Véhicules connectés.

Abstract

The Internet of Things (IoT) is defined as a complex infrastructure made up of a large number of heterogeneous devices. It relies on information and communication technologies to interconnect these devices and allow them to exchange their data to provide a variety of value-added services. The Web of Things (WoT) makes it possible to implement this vision by leveraging the Web to virtualize and interconnect IoT objects in a transparent way. IoT systems often operate in dynamic, unpredictable and mobile environments with limited resources. Therefore, virtualized IoT objects on the web need autonomous reasoning and decision-making skills to adapt their behaviors to their context. For this, we propose to extend passive virtual objects with reasoning mechanisms based on semantic models. This extension is called an *autonomous avatar*.

An avatar can provide multiple services which can be discovered, retrieved, and composed to create new IoT applications through service discovery and selection techniques. However, with the proliferation and exceptional growth in the number of connected objects, these methods are becoming problematic. To overcome these challenges, we offer the integration of social media concepts for IoT. They allow the collective intelligence to emerge from a group of avatars who can provide much more interesting and rich answers to complex problems than a single avatar. In addition, they accelerate and facilitate navigability in a large-scale dynamic IoT network. To integrate this concept, we propose an approach based on the notion of social distance. To reduce discovery search space and better target the transmission of requests to appropriate avatars, a clustering method based on the Fuzzy C-means algorithm is proposed to classify social avatars according to their functionality.

The discovered IoT services have different non-functional characteristics, particularly in terms of quality of service (QoS). In this context, we propose a new approach of distributed selection which takes into account QoS parameters as well as their fluctuation. A first approach based on a Multi-Objective Evolutionary Algorithm (MOEA) has been proposed in order to decompose the global QoS constraints into a set of local constraints thus making it possible to carry out a local selection. This approach was then extended by supervised machine learning mechanisms to predict local constraints. To do this, we have proposed an approach based on SVM (Support Vector Machine).

The proposed approaches are evaluated in the context of connected vehicles and through a series of experiments. The experimental results show the effectiveness of our approaches for the discovery and selection of services and the advantages they provide, in particular compared to traditional centralized approaches.

Key words : IoT, Avatar, Semantics, Social networks, Clustering, QoS, MOEA, SVM, Connected vehicles.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'Internet est un phénomène qui a significativement transfiguré notre société depuis sa création en 1990 [Vautherot, 2007]. Il s'agit d'un regroupement international d'une multitude de réseaux informatiques (domestiques, universitaires, commerciaux, gouvernementaux, etc) qui échangent de l'information par l'intermédiaire d'une série de protocoles de communication (TCP-IP) et qui donnent accès à plusieurs services et ressources, tels que : le streaming vidéo, le Web et la messagerie électronique (e-mail). Grâce au succès de l'Internet, la vie quotidienne de l'être humain est considérablement améliorée. Cette révolution technologique ne cesse de se renouveler, conduisant ainsi à l'émergence d'un nouveau paradigme appelé Internet des Objets (Internet of Things, IoT) permettant non seulement aux ordinateurs et aux différents équipements informatiques d'être connectés, mais également aux objets de la vie quotidienne, tels que : les véhicules et les bâtiments. Il permet aussi à ces objets d'interagir entre eux et d'échanger leurs données afin de coopérer et réaliser des tâches communes facilitant de plus en plus la vie humaine dans ses divers aspects. L'IoT est la révolution technologique la plus importante de l'Internet et du siècle. Il touche tous les domaines : la santé, la domotique, l'industrie, le transport, etc.

Au fil du temps, l'IoT a souffert d'une fragmentation verticale des approches adoptées pour couvrir les besoins des différents domaines d'application. En effet, diverses solutions avec des APIs distinctes ont été conçues séparément pour les différentes catégories d'applications. Cela a engendré des solutions silos uniques qui fonctionnent souvent sur le matériel spécifique du fournisseur ce qui limite considérablement le développement des applications IoT en les rendant très dépendantes du matériel et par conséquent, très difficiles à déployer et à maintenir. Pour pallier à ce problème, une nouvelle couche abstraire est conçue au-dessus de ce paradigme pour cacher tous les problèmes d'hétérogénéité IoT, soit au niveau des données ou soit au niveau des communications. Ceci a donné lieu au Web des Objets (Web of Things, WoT). Le WoT vise essentiellement à réduire le processus de développement des applications IoT en représentant chaque objet IoT (appareil physique ou entité logicielle) par un artefact sur le Web. Ces artefacts permettent d'exposer les fonctionnalités offertes par les divers dispositifs IoT sous forme de ressources accessibles via des interfaces standards.

Le WoT devrait intégrer un très grand nombre d'objets hétérogènes et omniprésents caractérisés par des interactions intenses créant ainsi des milliards de services. D'après une étude menée par le cabinet IoT Analytics, le déploiement de paradigme s'accroît à une vitesse exponentielle avec la perspective d'atteindre 22 milliards d'objets connectés en 2025 [Analytics, 2018]. Le WoT vise à créer une infrastructure globale qui fournit une variété de services à valeur ajoutée et des applications distribuées tirant parti des différentes fonctionnalités fournies par les divers objets intelligents composant cet

environnement. Dans ce contexte, la localisation d'un service spécifique est un défi crucial notamment à cause de l'hétérogénéité considérable et de la grande échelle du système.

Ces dernières années, beaucoup d'attention a été donnée au sujet de l'intégration des concepts de sociabilités dans le domaine du WoT donnant ainsi lieu au paradigme du Web Social des Objets (Social Web of Things, SoWoT) permettant ainsi de transformer les objets IoT en objets sociaux. Ces derniers ont ainsi la capacité de créer des liens sociaux entre eux d'une manière autonome pour faire preuve d'un comportement collectif dans la résolution des problèmes complexes. Cette nouvelle vision d'objets sociaux présente plusieurs avantages et promet d'importants progrès notamment en terme de découverte opportuniste des services IoT et en terme d'évolutivité des systèmes IoT via l'intégration de plus en plus d'objets connectés. De nombreuses recherches ont confirmé que les réseaux sociaux représentent un support extrêmement intéressant qui permet de faire émerger l'intelligence collective [Mennis, 2006] d'un groupe d'individus qui peuvent fournir des réponses beaucoup plus intéressantes et précises à des problèmes complexes qu'un seul individu.

Le développement des applications SoWoT répondant à des problèmes complexes repose sur la corrélation et la composition d'une multitude de capacités issues de plusieurs objets sociaux répartis géographiquement. Le paradigme de calcul orienté service (Service Oriented Computing, SOC) est utilisé pour supporter le développement léger, rapide et peu coûteux de ce genre d'applications en se basant sur le concept de service. Les services désignent des composants ouverts, auto-descriptifs et indépendants des plateformes de déploiement. Ils peuvent être décrits formellement, publiés dans des référentiels centraux ou d'une manière distribuée, découverts et assemblés les uns avec les autres pour mener un comportement collaboratif des objets fournissant ces services.

Très peu de contributions se sont intéressées au sujet de la découverte et la composition des services fournis par les objets sociaux dans l'infrastructure SoWoT. La plupart des approches proposées sont de nature centralisée ce qui les rend faillibles si une panne survient au niveau du contrôleur central. En plus, avec l'explosion de nombre d'objets connectés, ces méthodes ne peuvent pas surmonter les problématiques relatives au passage à large échelle. En ce qui concerne les approches décentralisées existantes, peu de détails sur les architectures et les modèles sociaux sont fournis. En plus, très peu d'entre eux se sont préoccupés des méthodes de découverte et de sélection de services d'une manière distribuée. Par conséquent, il est nécessaire de proposer de nouveaux modèles architecturaux qui répondent aux problèmes d'hétérogénéité d'un côté et assurent un comportement autonome, collaboratif, et social des objets intelligents de l'autre côté. Il est nécessaire également de trouver des solutions de découverte et de sélection des services fournis par les différents objets sociaux d'une manière distribuée et évolutive.

Problématiques & Défis

La mise en place d'une approche de découverte et de sélection distribuée des services IoT fournis par les objets intelligents et sociaux recouvre de nombreux défis et exigences. Dans ce travail, nous nous sommes concentrés sur les problématiques liées à l'hétérogénéité, la dynamicité et l'incertitude des systèmes IoT, l'évolutivité exponentielle de nombre des objets connectés, la sensibilité et l'adaptation au contexte d'utilisation ainsi que l'intégration des paramètres de qualité de service (Quality of Service, QoS).

Hétérogénéité

Les applications IoT sont basées sur des dispositifs et des technologies très hétérogènes issus de plusieurs constructeurs et utilisent des protocoles de communication très variés, ce qui soulève le problème d'interopérabilité et par conséquent, impose une fragmentation verticale importante du marché de l'IoT/WoT. Il existe deux types d'interopérabilité :

1. Interopérabilité syntaxique : elle est souvent obtenue par l'utilisation de normes pour les formats de données, les architectures, les interfaces ou même les protocoles d'échange [Aïssaoui, 2018]. Grâce à ce type d'interopérabilité, les systèmes IoT bénéficient de plusieurs services tels que : la découverte automatique d'une manière transparente de nouveaux appareils branchés sur le système, la surveillance des dispositifs, leur gestion, etc [Alaya, 2014]. Parmi les normes standardisées, on trouve le standard oneM2M [1]. Il vise à garantir l'interopérabilité au niveau de l'architecture, des formats de données ainsi que des protocoles utilisés.
2. Interopérabilité sémantique : elle a pour but de fournir des données significatives et compréhensibles par les machines. Elle permet de masquer l'hétérogénéité existante entre les divers objets communicants en utilisant des formalismes de représentation des connaissances proposées par le W3C (World Wide Web Consortium), et par conséquent, d'assurer une cohérence des données de bout en bout entre les applications IoT de haut niveau qui ont souvent une représentation complexe des données (valeur, unité, etc) [Seydoux, 2018].

Dynamicité et incertitude

Les environnements IoT sont par défaut de nature dynamique et incertaine. Ils sont caractérisés par des changements aléatoires et imprévisibles. Ces caractéristiques sont essentiellement dues à la mobilité des nœuds IoT, de leur apparence et de leur disparition d'une manière imprédictible et instable. Ces changements dynamiques de contexte

1. https://www.onem2m.org/

influent considérablement sur la fourniture des services et diminuent leur qualité. Cela peut aller d'une simple indisponibilité temporaire d'un service donné jusqu'à un dysfonctionnement et un arrêt total de son fonctionnement. Par conséquent, les applications IoT doivent être conscientes de ces modifications en permanence afin d'adapter leur comportement à ces changements pour assurer la continuité de leurs services.

L'évolution constante des besoins utilisateurs, les ressources matérielles disponibles, l'indisponibilité de certains services, les défaillances et les pannes physiques et logicielles constituent des exemples réels de ces changements.

Évolutivité et scalabilité

Au cours des dernières années, l'IoT a évolué à une vitesse exceptionnelle permettant ainsi d'interconnecter un nombre très important d'objets hétérogènes (capteurs, actionneurs, smartphones, applications, etc). L'évolution rapide du nombre d'objets connectés contribue au déploiement d'un système IoT scalable et omniprésent à large échelle. La gestion de tel système est très complexe et exige beaucoup d'effort et de ressources ce qui nécessite la mise en place des mécanismes de découverte et de sélection évolutifs.

Contexte et sensibilisation à la QoS

Les services IoT sont définis comme des modules logiciels qui regroupent un ensemble de fonctionnalités et qui peuvent être invoqués à distance indépendamment des plateformes de développement et de déploiement. Ces services possèdent également des propriétés non-fonctionnelles exprimées généralement à travers la QoS. La QoS représente un ensemble de caractéristiques qui mesurent la capacité des services à accomplir leurs fonctionnalités dans des états satisfaisants selon différentes métriques, par exemple, le coût temporel, la disponibilité, le coût monétaire, etc. Ces paramètres sont considérés comme des critères de choix pertinent dans le processus de sélection des services composant l'application à mettre en place. Cependant, ce processus est particulièrement problématique essentiellement, à cause, de nombre important d'objets candidats et la non-stabilité de leurs paramètres de QoS dans le temps.

À notre connaissance, il n'existe pas encore un travail qui réponde à toutes les exigences susmentionnées dans le contexte du SoWoT. Dans le cadre de cette thèse, nous souhaitons concevoir et implémenter une architecture distribuée basée sur des artefacts virtuels sur le Web, appelés *avatars*, qui permettent de représenter les objets IoT d'une manière sémantique et uniforme et les doter d'un comportement autonome et intelligent. En se basant sur cette architecture, une nouvelle approche distribuée pour la découverte des services IoT est proposée. Elle est fondée sur un modèle collaboratif qui exploite les relations sociales créées entre les objets intelligents pour

résoudre des problèmes complexes. Des algorithmes de clustering permettent de bien structurer l'espace de recherche et de découverte sont utilisés. Une fois la phase de découverte achevée, une approche de sélection distribuée basée sur la fluctuation de QoS est proposée pour désigner et choisir les services à intégrer dans la composition finale pour la résolution de l'objectif à accomplir. Cette approche est fondée sur le principe de décomposition des contraintes de QoS globales en contraintes locales. Elle est d'abord étudiée via une méthode basée sur un algorithme génétique évolutionnaire multi-objectifs (A Multi-Objective Evolutionary genetic Algorithm, MOEA), qui repose sur la collaboration d'un ensemble d'avatars. Cette approche a été ensuite étendue par une méthode basée sur la régression par machine à vecteurs de support (Support Vector Machine, SVM) pour la prédiction des contraintes de QoS locales. Cette dernière approche permet de prédire les contraintes locales en s'affranchissant les collaborations des avatars.

Cas d'utilisation : Dépassement des véhicules connectés

Cette section sera dédiée à la présentation du scénario adopté dans notre travail.

Contexte : le projet eHorizon

Cette thèse s'inscrit dans le cadre d'un projet de recherche et développement (R&D) nommé *eHorizon*, un projet sur la voiture autonome et connectée, issu d'une collaboration entre la filiale Continental Digital Services France de l'entreprise Continental Automotive de Toulouse et le laboratoire de recherche LAAS-CNRS. Ce projet est financé par l'État français, le conseil régional Occitanie et Toulouse Métropole. Il se positionne dans un contexte de ville intelligente et de déploiement d'objets connectés dans les infrastructures routières. Dans ce contexte, de nombreuses opportunités, qui visent à créer de nouvelles solutions intelligentes et de nouveaux scénarios innovants autour des véhicules connectés du futur, se présentent.

Les systèmes de transport intelligents sont définis comme une combinaison des technologies de communication, d'information et du positionnement géographique, destinés principalement à améliorer l'expérience de conduite et rehausser la sécurité de transport routier. Ils permettent d'automatiser les activités exigeant l'intervention humaine et la gestion de la circulation routière. Afin de mettre en place des systèmes complexes au sein du domaine de transport intelligent, plusieurs technologies émergentes et diverses disciplines scientifiques sont employées notamment les objets connectés avec leurs différents types et les diverses plateformes de services. Les véhicules connectés doivent exploiter efficacement ces technologies pour réaliser des expérimentations ambitieuses dans l'infrastructure routière d'une manière intelligente et collaborative. Actuellement,

de nombreux véhicules sont connectés et ils sont équipés de plusieurs objets connectés qui leur permettent de collecter, traiter et diffuser des informations utiles pour accomplir plusieurs scénarios et applications routières complexes. Le scénario de dépassement représente un exemple qui reflète bien l'ensemble des objectifs cités ci-dessus. Par conséquent, pour bien illustrer le fonctionnement de nos approches de modélisation, de découverte et de sélection des services IoT fournis par les objets connectés hétérogènes, nous avons adopté ce cas d'utilisation. Ce choix de scénario n'a pas de conséquence sur la généricité de nos propositions qui peuvent être appliquées dans d'autres scénarios ou d'autres domaines comme la domotique et l'agriculture.

Description fonctionnelle du scénario de dépassement

Le dépassement représente le fait que deux véhicules, qui circulent dans le même sens de circulation et la même voie, se dépassent. Ce scénario s'avère complexe et plusieurs conditions de circulation doivent être satisfaites pour rendre le dépassement plus sécurisé et éviter toutes les circonstances dangereuses qui peuvent représenter des risques pour le conducteur et les autres véhicules. Il s'effectue généralement par la gauche et avant de dépasser, le conducteur doit vérifier et surveiller les véhicules venant de l'arrière ou qui se trouvent dans l'angle mort gauche en utilisant les rétroviseurs de son véhicule. Lorsque le moment est approprié, il allume le clignotant avant de changer de voie. Lors de changement de voie, le conducteur doit choisir la vitesse la plus adéquate en respectant les vitesses maximales autorisées sur la route sur laquelle se trouve le véhicule tout en respectant la distance de sécurité vers l'avant. La Figure 1 donne un aperçu global sur cette manœuvre [ornikar, 2020].

FIGURE 1 – Manœuvre de dépassement des véhicules

Dans ce scénario, le véhicule hôte, c'est-à-dire celui qui cherche à dépasser, n'est pas équipé de tous les dispositifs IoT requis qui lui fournissent la visibilité nécessaire pour effectuer la manœuvre de dépassement tout seul d'une manière sécurisée. Par

conséquent, nous visons à travers ce scénario de permettre à ce véhicule de chercher (découvrir) et à sélectionner les appareils IoT appropriés qui peuvent collaborer avec lui pour accomplir le dépassement. De plus, les objets IoT découverts offrent différentes valeurs de QoS et le processus de dépassement peut être désormais soumis à des exigences globales de QoS de bout en bout. Notre objectif est donc de sélectionner une combinaison parmi les objets découverts qui répondent aux contraintes de QoS globales sans examiner toutes les combinaisons possibles d'objets.

Grandes lignes de la thèse

Ce document est structuré en cinq chapitres comme suivant :

Chapitre 1 : Contexte scientifique & État de l'art

Ce chapitre comporte essentiellement une étude des principaux concepts couverts par cette thèse. La première partie présente une vision globale de l'IoT et du WoT, leurs technologies et les protocoles récurrents dans ces domaines. Dans la deuxième partie, les aspects relatifs au développement des applications distribuées en rapport avec les systèmes orientés services, le Web sémantique, les systèmes multi-agents et les technologies de déploiement cloud/fog computing sont détaillés. Et enfin, un état de l'art sur la découverte et la sélection des services est élaboré afin d'étudier l'existant et analyser les problèmes des travaux antérieurs afin de proposer des solutions adaptées.

Chapitre 2 : Vers une architecture IoT distribuée basée sur des avatars autonomes

Dans le second chapitre, la première contribution est détaillée. Elle consiste en une architecture modulaire basée sur la notion d'avatar autonome pour la fourniture des services à valeur ajoutée. L'idée de cette proposition est de doter les avatars, qui sont des doubles virtuels des entités physiques ou logicielles sur le Web, d'un comportement autonome. Cette démarche permet de rendre les avatars actifs et capables de prendre des décisions intelligentes en fonction de leur contexte. De plus, doter les avatars d'un raisonnement sémantique intelligent offre l'opportunité de dépasser les limites des approches existantes qui reposent sur un contrôleur central qui manipule et gère les avatars passifs. En effet, les approches centralisées ne sont pas évolutives et dans le cas d'un dysfonctionnement, l'ensemble du système peut être compromis.

Chapitre 3 : Vers une découverte distribuée basée sur les réseaux sociaux et le clustering

Le troisième chapitre, quant à lui, présente une nouvelle approche distribuée pour la découverte des services fournis par les avatars en exploitant les relations sociales (intérêts, location, etc) qui peuvent exister entre eux afin de leur permettre de

collaborer et d'accomplir des processus applicatifs complexes. Cette approche utilise également un algorithme de clustering pour classifier les avatars du réseau social à exploiter selon leurs fonctionnalités afin de guider au mieux la transmission des demandes de découverte aux avatars appropriés.

Chapitre 4 : Vers une sélection distribuée basée sur le MOEA et la SVM

Ce quatrième chapitre est consacré à la présentation de nos deux dernières contributions concernant la sélection distribuée des services IoT découverts via la méthode proposée dans le chapitre précédant. Cette proposition permet de garantir la fiabilité de la composition résultante en considérant la fluctuation de QoS des services IoT considérés. La première contribution est basée sur la décomposition des contraintes de QoS globales en contraintes locales qui servent de seuils supérieurs/inférieurs, pour la sélection locale et parallèle des services dans les différents Clusters, en utilisant un algorithme génétique évolutionnaire multi-objectifs (MOEA). En ce qui concerne notre deuxième contribution, elle permet la prédiction des contraintes de QoS locales en se basant sur les expériences antérieures des avatars via la méthode de machine à vecteurs de support (SVM) qui fait partie des méthodes d'apprentissage automatique supervisé (supervised machine learning).

Chapitre 5 : Évaluation expérimentale

Ce dernier chapitre est consacré aux évaluations des approches de découverte de sélection des services IoT proposées dans ce travail de thèse. Tout d'abord, l'architecture technique et l'environnement de réalisation du système et les principaux choix techniques sont présentés. Ensuite, dans un premier temps, nous évaluons la méthode de découverte des services IoT en analysant son temps de réponse et son taux de réussite (success rate) en faisant varier ses paramètres relatifs afin de caractériser la qualité des résultats produits. Après cela, les méthodes de MOEA et SVM pour la sélection distribuée des services sont évaluées essentiellement en terme d'optimalité et de temps de réponse.

À travers ces différents chapitres, le comportement fonctionnel du système est expliqué et évalué à travers un scénario de dépassement dans le cadre des véhicules connectés.

Ce document se termine par une conclusion générale dans laquelle nous récapitulons les différents résultats auxquels nous avons abouti au cours de notre travail ainsi que nos perspectives dans ce domaine.

Chapitre 1

Contexte scientifique & État de l'art

Sommaire

Le développement des applications IoT distribuées à large-échelle touche à plusieurs technologies innovantes relatives principalement à la description, la structuration et la gestion des données fournies par les objets connectés hétérogènes et leurs fonctionnalités ainsi que leur déploiement dans les architectures distribuées.

Pour cela, nous visons à travers ce chapitre à introduire les concepts principaux de l'IoT et de WoT ainsi que leurs caractéristiques fondamentales. Les aspects relatifs au développement des applications distribuées en rapport avec les systèmes orientés services, le web sémantique et les systèmes multi-agents pour comprendre leur nature autonome et distribuée et ce qu'ils apportent pour l'IoT, sont ensuite détaillés. Après cela, les technologies fog/cloud computing nécessaires au déploiement à large échelle des systèmes IoT sont analysées. Enfin, avant de conclure ce chapitre, un état de l'art sur les méthodes de découverte et de sélection des services est élaboré à travers la présentation des principales contributions dans les domaines des services Web traditionnels, des systèmes multi-agents et l'IoT.

1.1 Généralités sur l'IoT et le WoT

1.1.1 Internet des Objets (IoT)

Le terme de l'Internet des Objets ne dispose pas d'une définition standard, unifiée et partagée. Plusieurs définitions de ce paradigme ont été proposées dans la littérature.

L'Union internationale des télécommunications définit l'IoT comme étant : « une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution » [ITU-T, 2012].

En réalité, ce concept possède une dimension conceptuelle et une autre technique. D'un point de vue conceptuel, nous désignons par l'IoT : « des objets ayant des identités et des personnalités virtuelles, opérant dans des espaces intelligents et utilisant des interfaces intelligentes pour se connecter et communiquer au sein de contextes d'usages variés » [Duce, 2008].

En ce qui concerne la dimension technique, la définition émergente donnée de l'IoT est donc : « un réseau de réseaux qui permet, via des systèmes d'identification électronique normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi de pouvoir récupérer, stocker, transférer et traiter, sans discontinuité entre les mondes physiques et virtuels les données s'y rattachant » [Benghozi et al., 2008].

D'après les définitions citées ci-dessus, nous pouvons définir l'IoT comme : un pa-

radigme qui relie divers objets physiques (capteurs, montres, véhicules, etc.) et virtuels (logiciels, applications, systèmes, etc.) par l'intermédiaire d'une infrastructure réseau existante. Cette dernière offre la possibilité à ces objets de communiquer et d'échanger les données nécessaires à leur fonctionnement. Chaque objet est identifiable d'une manière unique grâce à un système électronique embarqué (RFID, adresse, etc.). Cela permet à ces objets d'être détectés et contrôlés à distance d'une manière simple et non-ambiguë.

1.1.2 Web des Objets (WoT)

Il n'existe pas encore une définition unique du WoT. La communauté WoT [Webofthings, 2016] a donné la définition suivante pour le WoT : « le Web des Objets est un paradigme qui a pour objectif de créer l'Internet des objets d'une manière véritablement ouverte, flexible et évolutive, en utilisant le Web comme couche applicative ».

Wenyi Xu a donné dans sa thèse [Xu, 2015] une description assez complète de cette technologie : « le WoT peut être considéré comme un IoT concerné par le Web et les technologies connexes. Le Web est une plate-forme qui permet la communication globale entre les différents utilisateurs. L'IoT permet de connecter physiquement les objets et leurs données, tandis que le WoT utilise le Web comme plate-forme où les objets et leurs données peuvent être représentés et les informations qu'ils génèrent peuvent être échangées. L'hétérogénéité des objets et leurs informations peuvent être utilisées dans différentes applications. La coopération d'objets permet de s'adapter les uns aux autres ou de fournir de nouveaux types d'informations ».

Quant à la communauté de standardisation W3C, une définition formelle est donnée dans leur document [W3C, 2016]. Le WoT est défini comme « une infrastructure qui vise à permettre l'interopérabilité entre les plateformes et applications des domaines IoT. Il fournit en premier des mécanismes pour décrire formellement les interfaces IoT afin de permettre aux périphériques et aux services IoT de communiquer les uns avec les autres, indépendamment de leur implémentation sous-jacente et via plusieurs protocoles de réseau. Deuxièmement, il fournit un moyen normalisé de définir et de programmer le comportement IoT ».

À partir de ce qui a été présenté dans les sous-sections 1.1.1 et 1.1.2, nous pouvons constater que le WoT n'est qu'une couche abstraite au dessus de l'IoT qui permet d'assurer l'interopérabilité des objets IoT hétérogènes.

1.1.3 Architecture IoT/WoT

Dans un premier temps, il convient d'élaborer la vue architecturale sur laquelle s'appuie le paradigme de l'IoT pour mieux comprendre son fonctionnement. Nous re-

marquons dans la Figure 1.1 que la structure de l'IoT est divisée en cinq couches décrites comme suit [Zaheer and al., 2012].

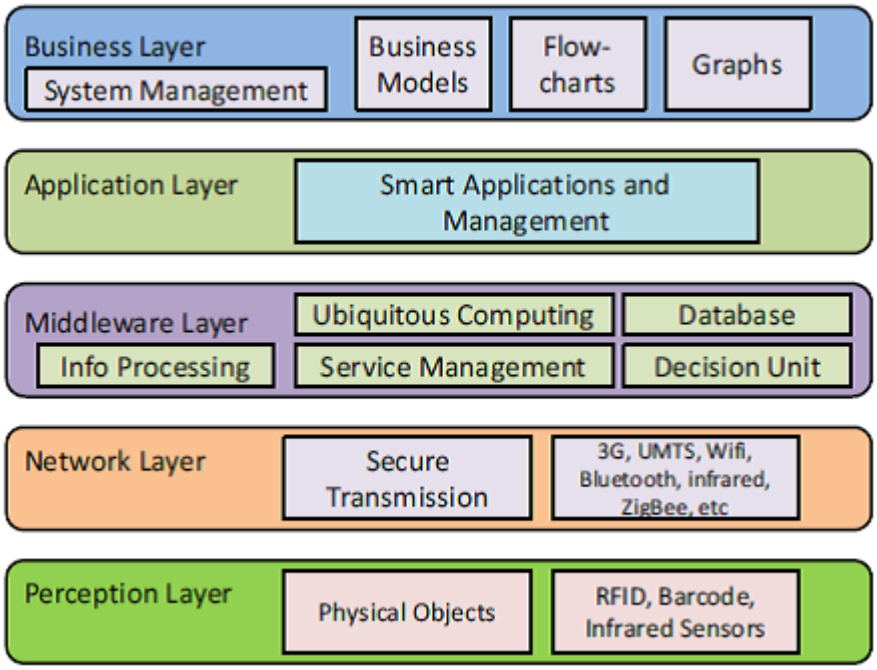


FIGURE 1.1 – Architecture IoT/WoT en couches, [Zaheer and al., 2012]

- **La couche perception** : elle regroupe les divers dispositifs (les lecteurs, les différents types de capteurs, etc.) qui ont pour fonction de capter les grandeurs physiques analogiques produites par les autres objets connectés telles que l'humidité et le mouvement. Ces informations sont ensuite traitées et transmises vers la couche réseau.
- **La couche réseau** : appelée aussi couche de transmission. Elle a pour but de router les informations captées par la couche perception vers la couche middleware d'une manière sécurisée par le biais d'un support filaire ou non-filaire (WIFI, Bluetooth, etc.) selon la nature du capteur.
- **La couche middleware** : elle sert d'interface entre la couche matérielle et la couche application. Elle reçoit des informations de la couche réseau, les stocke dans une base de données pour leur appliquer des traitements et des calculs pour aboutir à de nouvelles informations. Cette couche n'est que la couche d'intéropérabilité WoT.
- **La couche applicative** : c'est la couche responsable de la livraison des différentes applications aux utilisateurs dans le cadre de l'IoT. Ces applications peuvent

provenir de plusieurs domaines tels que le transport, la domotique, l'agriculture, etc.

- **La couche affaire (Business)** : elle a pour but de mettre en place des modèles d'entreprises, des graphiques et des organigrammes sur la base des données reçues de la couche immédiatement inférieure (applicative). Ces modules sont utilisés pour gérer le système IoT d'une manière efficace.

1.1.4 Technologies IoT/WoT

La mise en place des concepts de l'IoT et du WoT a vu le jour grâce à plusieurs technologies et en s'appuyant sur une série de protocoles de l'Internet. Ces technologies se situent au niveau des couches les plus basses de l'architecture de l'IoT (la couche perception, réseau et middleware). Parmi eux, nous pouvons citer les technologies suivantes :

- Réseaux d'accès : c'est l'ensemble des moyens utilisés pour relier les équipements de télécommunication à l'infrastructure réseau. Par exemple : IEEE 802.15.4, Zigbee et 6LoWPAN.
- Protocoles de communication applicatifs : c'est l'ensemble des procédures suivies pour l'envoi et la réception des données sur un réseau. Par exemple : HTTP, CoAP et MQTT.
- Sérialisation de données : ce sont les méthodes utilisées pour structurer les données échangées au sein d'un réseau dans un format spécifique afin de faciliter leur traitement. Par exemple : XML et JSON.
- Identification des dispositifs : c'est l'ensemble des technologies assurant l'identification et la traçabilité des divers objets issus de l'IoT. Par exemple : RFID et NFC.

1.2 Systèmes orientés services

L'architecture logicielle d'un système informatique donné permet de décrire d'une manière structurelle les composants logiciels élémentaires qui le composent, leurs propriétés fonctionnelles et non-fonctionnelles, ainsi que les interactions qui peuvent se produire entre eux. L'architecture orientée service (Service-Oriented Architecture, SOA) [Figer, 2005] est un paradigme d'ingénierie interdisciplinaire. Il est basé sur un ensemble de composants résultant de la décomposition des fonctionnalités complexes en un ensemble de fonctions simples connues sous le nom de *services*. Cette approche est un choix répandu dans l'Internet d'aujourd'hui et se révèle très adaptée au paradigme de l'IoT grâce à sa conformité, son utilisation de HTTP et sa légèreté. Des messages aux contenus légers s'avèrent adéquats aux besoins des capacités limitées des objets connectés.

1.2.1 Architecture orientée service (SOA)

L'architecture SOA fournit une vision pour la construction et l'utilisation des applications logicielles distribuées. Plusieurs définitions techniques et métiers sont données à cette technologie.

Selon Sarang et al. [Sarang, 2007], « une architecture SOA est une structure d'intégration de processus métier qui supporte une infrastructure des technologies d'information comme étant composants, services sécurisés, standardisés et qui peuvent être combinés pour s'adresser aux priorités de changements métiers ».

Alors que Juric et al. [Juric, 2007] ont définit la SOA comme : « un style d'architecture logicielle qui consiste à concevoir des applications et des systèmes distribués pour fournir des services métier dotés d'interfaces auto-descriptives bien définies et qui sont utilisés pour composer les processus métiers de l'entreprise ».

De ces définitions précédentes, nous pouvons constater que l'architecture SOA est une évolution de l'architecture client/serveur en conjonction avec le domaine des systèmes distribués.

L'architecture de référence de la SOA repose sur trois composants comme le montre la Figure 1.2.

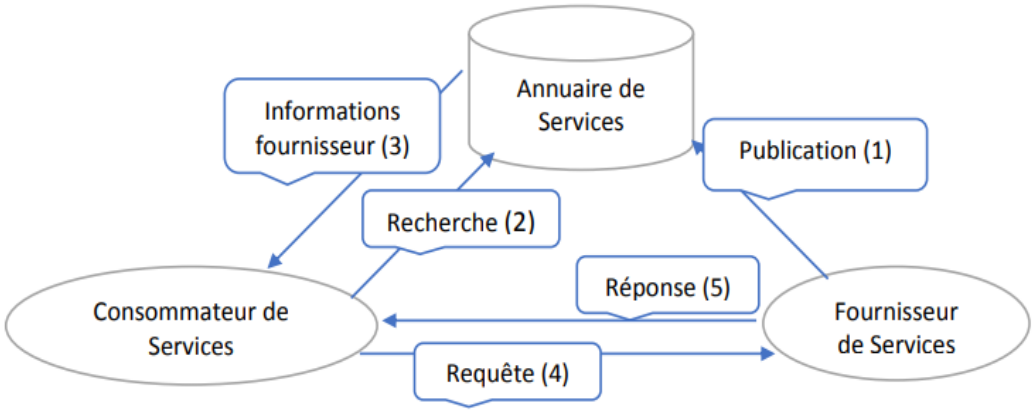


FIGURE 1.2 – Architecture Orientée Services (SOA) [Liu and Liu, 2009]

- Fournisseur de services : c'est une entité logicielle adressable sur le réseau qui offre des services bien définis. Elle peut recevoir, négocier et exécuter les requêtes des consommateurs.
- Consommateur de services : il s'agit de l'application cliente qui va rechercher et demander des services particuliers.
- Annuaire de services (UDDI) : c'est un registre réseau où les fournisseurs peuvent

publier les descriptions des services qu'ils offrent et les consommateurs recherches les services requis.

Le processus de publication et de découverte des services est accompli selon les étapes suivantes :

1. Un fournisseur publie la description de ses services dans l'annuaire UDDI.
2. Un consommateur de services consulte l'annuaire pour rechercher le service dont il a besoin.
3. L'annuaire localise le service demandé et oriente le consommateur vers le fournisseur en lui envoyant l'adresse de ce dernier.
4. Le consommateur envoie sa requête au fournisseur afin de demander le service.
5. Le fournisseur répond à la requête du consommateur et lui retourne les résultats attendus.

1.2.2 Services : définition et modélisation

Plusieurs définitions sont données aux services Web.

Le W3C définit un service Web comme [W3C, 2004] : « un système logiciel conçu pour prendre en charge l'interaction interopérable de machine à machine sur un réseau. Il a une interface décrite dans un format exploitable par une machine (spécifiquement WSDL). D'autres systèmes interagissent avec le service Web d'une manière prescrite par sa description à l'aide de messages SOAP, généralement transmis à l'aide du HTTP avec une sérialisation XML conjointement avec d'autres normes liées au Web ».

D'après IBM [IBM, 2001], « les services Web sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le Web. Les services web effectuent des actions allant de simples requêtes à des processus métiers complexes. Une fois qu'un service web est déployé, d'autres applications (y compris des services web) peuvent le découvrir et l'invoquer ».

À travers les définitions données ci-dessus, nous pouvons définir un service Web comme un module logiciel qui regroupe un ensemble de fonctionnalités qui peuvent être invoquées à distance indépendamment des plateformes de développement et de déploiement.

Description des services

Plusieurs standards et langages ont été proposés dans la littérature pour la description des services Web. Ces descriptions consistent principalement à définir d'une manière interopérable et compréhensible les différentes interfaces qui exposent les diverses fonctionnalités fournies par les services et leurs exigences non-fonctionnelles.

Il existe principalement deux modèles de description : modèles syntaxiques et sémantiques. Nous commençons tout d'abord par détailler les descriptions syntaxiques qui sont largement utilisées vu leur simplicité et leur efficacité. Par la suite, nous présentons les principaux modèles sémantiques.

- Modèles syntaxiques : ce sont des formalismes standards à appliquer pour donner et structurer les descriptions des services. Parmi ces formalismes :

- Web Service Description Language (WSDL) [Christensen et al., 2001] : c'est un standard largement recommandé par le W3C qui est basé sur une grammaire reposant sur la notation XML. Il permet de décrire principalement l'emplacement du service, ses opérations en terme d'entrées et sorties, ses données et leur type ainsi que ses méthodes d'invocation.
- Web Service Policy (WS-Policy) [Weerawarana et al., 2005] : ce langage fait partie de WS* Spécifications et il permet de décrire d'une manière flexible et extensible les différentes entités composant un système à base de services en tant que policies.

- Modèles sémantiques : ils permettent de fournir des descriptions compréhensibles par une machine. La sémantique a pour but de masquer l'hétérogénéité existante entre les divers services communicants en utilisant des formalismes de représentation des connaissances standards. Parmi ces standards :

- DARPA agent markup language for services (DAML-S) [Ankolekar et al., 2001] : il s'agit d'un langage basé sur l'ontologie DAML-S qui permet aux agents logiciels et aux utilisateurs de découvrir, d'invoquer et de composer facilement les services Web particuliers selon leurs caractéristiques fonctionnelles et non-fonctionnelles. Cette ontologie comprend trois parties :
  - Service profile : il définit les caractéristiques globales du service et ses paramètres de localisation.
  - Process Model : pour décrire le comportement et le fonctionnement détaillé du service considéré.
  - Grounding : pour décrire les détails d'interaction et d'invocation du service et les échanges avec lui via les messages.
- Web Service Description Language-Semantic (WSDL-S) [Akkiraju et al., 2005] : ce langage représente une extension du langage syntaxique traditionnel WSDL avec de la sémantique. Il consiste à utiliser les descriptions WSDL tout en référençant l'ontologie sémantique de service dans la partie définition et en ajoutant deux balises : Action et Contrainte pour annoter les opérations WSDL. La première balise ajoutée Action permet d'exposer l'acte de l'opération et la deuxième, ses pré et post conditions.
- Resource Description Framework (RDF) [Lassila et al., 1998] : c'est un modèle proposé par W3C pour la description standardisée et formelle des services Web

et leurs métadonnées à travers un graphe. Il est basé sur la notion de triplet composé de :

- Sujet : exprime la ressource à décrire.
- Prédicat : expose une propriété de cette ressource.
- Objet : décrit la valeur de la propriété qui peut être une donnée ou une autre ressource.

- Ontology Web Language for Services (OWL-S) [Martin et al., 2004] : c'est un langage d'ontologie basé sur le modèle sémantique OWL pour décrire les services web ainsi que leurs opérations et leurs métadonnées d'une manière formelle non-ambiguë. Il est essentiellement développé pour permettre l'automatisation de la découverte, d'invocation, la composition et le monitoring des ressources Web ayant des capacités et propriétés particulières.
- Web Service Modeling Ontology (WSMO) [Domingue et al., 2005] : ce langage fournit un cadre conceptuel basé sur le balisage sémantique des principaux aspects liés aux services Web afin de compléter les normes des descriptions syntaxiques existantes. Il permet également de prendre en charge les mécanismes de déploiement et d'interopérabilité en se basant sur quatre éléments :
  - Objectif : il représente l'objectif de l'utilisateur lors de l'invocation du service.
  - Ontologies : elles permettent de fournir les informations utilisées par tous les autres composants.
  - Médiateurs : ils représentent des connecteurs entre les différents composants et ont pour objectif d'assurer l'interopérabilité entre les différentes ontologies.
  - Web Service : il représente la description sémantique des capacités et des propriétés du service et des informations concernant l'utilisation de ses interfaces.

1.2.3 Types de services

Il existe deux types de Web Services : SOAP et REST. Nous allons détailler dans cette partie chaque type séparément.

- **SOAP (Simple Object Access Protocol) :** un service SOAP est basé sur le protocole de communication orienté objet SOAP permettant l'échange de messages en format XML entre un client (une application) et un fournisseur de services, utilisant généralement HTTP comme protocole de transfert de données. Il est indépendant de tout système d'exploitation et de tout langage de programmation [IBM, 2014] [Paik et al., 2017].

- **REST (Represntational State Transfer) :** un service REST est une mise en œuvre de l'architecture orientée ressources. Il est conçu selon une architecture client/serveur. Il est utilisé dans le but de faciliter le développement des applications orientées services via l'utilisation de HTTP comme un protocole de transport au lieu de SOAP qui pose problème à cause de la taille non-négligeable des messages utilisés sous format XML. Donc, au lieu d'utiliser HTTP juste comme simple transporteur de messages SOAP, on remplace SOAP par le protocole HTTP qui dispose aussi de tous les mécanismes mis en œuvre dans SOAP (les opérations CRUD, typage des données (Content-type) et les entêtes (headers), ainsi que l'accès aux ressources spécifiées dans l'URL) [Paik et al., 2017]. REST décrit la manière dont une ressource doit être manipulée via le protocole HTTP à l'aide de cinq méthodes principales :
  - Créer une nouvelle ressource (POST).
  - Récupérer la représentation d'une ressource (GET).
  - Supprimer une ressource (DELETE).
  - Modifier une ressource (PUT).
  - Obtenir des méta-informations d'une ressource (HEAD).

En général, l'utilisation de services SOAP demande une plus grande bande passante et l'encodage et le décodage des messages SOAP basés sur XML consomment plus de ressources que les services REST.

1.3 Émergence du Web Sémantique dans l'IoT

Le Web sémantique ou le Web 3.0 [Hendler, 2009] comme certains l'appellent n'est qu'une extension standardisée par le W3C du Web actuel qui vise à étendre les capacités de transfert de données de ce dernier pour permettre aux différentes entités du système de comprendre et d'interpréter les informations échangées.

Ce paradigme a initialement été proposé par Tim Berners-Lee, le lauréat du prix Turing 2016 [Gandon, 2017], durant l'un de ses communiqués en 1994 [Tim, 1994]. Il a exprimé sa vision avec la phrase suivante : « l'ajout de sémantique au Web implique deux choses : autoriser les documents contenant des informations sous des formes lisibles par la machine et autoriser la création de liens avec des valeurs de relation. Ce n'est que lorsque nous aurons ce niveau supplémentaire de sémantique que nous pourrons utiliser la puissance de l'ordinateur pour nous aider à exploiter l'information dans une plus grande mesure que notre propre lecture » [Berners-Lee et al., 2001]. Il modélise le Web comme un espace de connaissances qui relie et structure les données appelées données liées (Linked Data, LD) [Bizer et al., 2011] pour permettre l'invocation des différentes informations stockées dans le Web d'une manière facile et efficace.

Le Web sémantique permet principalement la représentation et l'ingénierie des connaissances. Il est composé des éléments suivants [Bach, 2006] :

- Des ressources identifiées d'une manière unique avec des URIs (Uniform Resource Identifier).
- Des liens logiques permettant de connecter les ressources entre elles.
- Des normes W3C pour la description standardisée et interopérable des ressources et leurs propriétés en se basant sur les ontologies.

La notion d'ontologie représente l'un des principaux concepts sur lesquels se base le Web sémantique [Charlet et al., 2004]. Elles représentent un moyen efficace pour structurer intelligemment un domaine, en fournissant une indexation de contenu des diverses ressources en utilisant des annotations sémantiques pour représenter les connaissances explicites relatives aux entités composant un domaine donné. Les ontologies servent à fournir un vocabulaire et un modèle sans ambiguïté sur les objets, leurs propriétés ainsi que leurs relations. Ce formalisme a été initialement introduit par Gruber [Gruber, 1991] qui l'a défini comme suivant : « une ontologie est une spécification explicite et formelle d'une conceptualisation d'un domaine de connaissance ». Telles qu'elles sont définies, les ontologies ont pour objectif principal d'automatiser la manipulation et le traitement des informations. Elles sont constituées généralement de :

- Plusieurs classes : elles correspondent aux abstractions des objets et des principaux concepts appartenant au domaine modélisé.
- Individus : ils représentent des instances des différentes classes d'ontologie.
- Attributs : ils représentent les différentes caractéristiques et propriétés relatives à un concept donné.
- Liens et relations : ils reflètent l'ensemble des rapports reliant les différents concepts de l'ontologie.
- Axiomes : elles constituent le fondement de raisonnement sémantique. Elles représentent des assertions acceptées comme vraies sans démonstration.

Un des principaux défis que le domaine de Web sémantique doit surmonter est la coexistence de plusieurs ontologies différentes qui se basent sur des vocabulaires distincts et hétérogènes. Pour cela, les mécanismes d'alignement d'ontologies [Safar and Reynaud, 2009] sont proposés. Ils visent essentiellement à identifier des similarités et les correspondances entre les concepts des ontologies à aligner et les normaliser via des vocabulaires standards.

1.3.1 Ontologies IoT/WoT

Plusieurs ontologies ont été proposées dans le cadre de l'IoT et du WoT qui visent à assurer une interopérabilité sémantique des objets connectés de bout en bout et une

découverte automatique des services. Parmi les ontologies les plus prometteuses de ces domaines, nous trouvons :

- IoT-O [1](Internet of Things Ontology) [Seydoux et al., 2016] : IoT-O est une ontologie modulaire proposant un vocabulaire commun, non-ambigu et compréhensible par les machines. IoT-O décrit sémantiquement les dispositifs connectés et leurs données afin de les rendre conscients de leur environnement. Pour concevoir IoT-O, un ensemble d'ontologies bien définies ont été soigneusement sélectionnées et importées dans cette ontologie comme SSN, SAN et MSM.
- SSN [2] (Semantic Sensor Network) [Compton et al., 2012] : pour les capteurs et les observations. Elle permet de représenter les propriétés et les capacités de mesure des capteurs pour fournir une description générique de ces dispositifs. Elle sert également à décrire les données collectées par les capteurs et le contexte de leur acquisition.
- SOSA [3] (Sensor, Observation, Sample, and Actuator) [Janowicz et al., 2019] : une nouvelle version de SSN qui enrichie et élargie son domaine d'utilisation en intégrant le concept d'actionneur et d'échantillon.
- SAN [4](Semantic Actuator Network) [Seydoux et al., 2016] : pour les actionneurs et les actions. Il s'agit d'une nouvelle ontologie inspirée de SSN pour représenter les diverses propriétés des actionneurs. Elle permet de saisir les connaissances que le système a de ses propres capacités d'impact sur son environnement et de le faire évoluer en modélisant la transformation des actions abstraites par les actionneurs en actions réelles.
- MSM [5](Minimal Service Model) [Pedrinaci et al., 2010] : pour décrire les services et la façon dont ils peuvent être sollicités sans ambiguïté afin de réduire la complexité aux utilisateurs comme les méthodes d'entrée et le contenu des messages de sortie.
- Lifecycle [6] [Ibrahim and Ataelfadiel, 2018] : pour décrire les états par lesquels les données, les appareils, les services et les composants IoT peuvent passer dans leur cycle de vie. Ce dernier est représenté par une évolution à travers un ensemble d'états discrets.
- PowerOnt [7] [Bonino et al., 2014] : la gestion de l'énergie est un sujet crucial dans les systèmes IoT. Cette ontologie sert à décrire les propriétés de consommation d'énergie des divers dispositifs.

1. https://www.irit.fr/recherches/MELODI/ontologies/IoT-O
2. http://www.w3.org/ns/ssn/
3. http://www.w3.org/ns/sosa/
4. https://www.irit.fr/recherches/MELODI/ontologies/SAN.owl
5. http://iserve.kmi.open.ac.uk/ns/msm
6. https://www.irit.fr/recherches/MELODI/ontologies/IoT-Lifecycle
7. http://elite.polito.it/ontologies/poweront.owl

- DUL [8] [Presutti and Gangemi, 2008] : ontologie de haut niveau qui décrit des concepts très généraux. Elle vise à faciliter l'interopérabilité entre les ontologies différentes. DUL est utilisée pour maximiser l'extensibilité, la réutilisation et faciliter alignement des diverses ontologies décrites précédemment.
- Autonomic [de Azevedo et al., 2010] : cette ontologie permet d'orienter les décisions prises par les systèmes autonomes selon les politiques de haut niveau définies par l'utilisateur et les données collectées par les objets.
- oneM2M base ontology [9] : l'ontologie oneM2M base Ontology est l'ontologie minimale spécifiée par le standard oneM2M qui représente le cœur de domaine IoT. Elle définit un ensemble minimal de concepts de haut niveau de telle sorte que n'importe quelle ontologie peut être mappée dans oneM2M [10].
- SAREF [11] (Smart Appliance REFerence) [Daniele et al., 2016] : cette ontologie est conçue initialement par l'ETSI pour le domaine de la domotique. Elle était dédiée essentiellement à la gestion d'énergie et la sécurité domotique. SAREF a été ensuite étendue pour couvrir les spécificités d'autres domaines IoT.
- WSMO (Web Service Modeling Ontology) [Roman et al., 2005] : initialement proposée par Digital Enterprise Research Institute (DERI), l'institut de recherche européen spécialisé dans les Web services sémantiques. Cette ontologie fournit un cadre conceptuel formel avec une approche top-down pour modéliser sémantiquement les comportements et les aspects non-fonctionnels des Web services dans le but d'automatiser leur découverte et leur invocation [Chhun et al., 2016] .
- hRESTs (HTML for RESTful Services) [Kopeckỳ et al., 2008] : elle représente un cadre de micro-annotations décrivant les différentes opérations des Web services REST pour permettre leur invocation automatique via des API Web. Cette ontologie s'appuie sur la documentation HTML [Maleshkova et al., 2009].

1.3.2 Raisonnement sémantique

Après avoir décrit et structuré les données du Web sémantique via des ontologies, des mécanismes de raisonnement et d'inférence sont utilisés pour permettre l'exécution et le traitement automatique des connaissances modélisées. L'inférence [McGuinness and Da Silva, 2004] est un mécanisme cognitif basé sur un processus déductif qui permet de produire (inférer) de nouvelles connaissances à partir des connaissances déjà acquises dans la base de connaissances. Ce mécanisme se base sur des raisonneurs implémentés avec la logique théorique souvent avec les langages RDF et OWL. Dans le processus de raisonnement, le moteur d'inférence applique des règles d'inférence

8. http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
9. https://git.onem2m.org/MAS/BaseOntology/raw/master/base_ontology.owl
10. http://www.onem2m.org/technical/published-drafts
11. http://ontology.tno.nl/saref/

logiques sur l'ensemble des connaissances disponibles dans la base de connaissances (KB). L'ensemble de règles de raisonnement utilisées par le moteur d'inférence sont définies par des hypothèses à valider pour déduire de nouveaux faits et connaissances.

Plusieurs formalismes ont été proposés pour modéliser les règles d'inférence, tels que : SHACL, SPIN et SWRL soumis par le W3C. SHACL (Shapes Constraint Language) [Knublauch, 2017] représente la nouvelle recommandation du W3C. Il s'agit d'un langage utilisé pour décrire et définir des contraintes sur le contenu des nœuds de graphe RDF en tant que forme. Il fournit un formalisme de haut niveau pour exprimer des conditions sur le type de données de certaines propriétés, leurs cardinalités et autres contraintes. Il peut également exprimer des contraintes complexes avec SPARQL et d'autres langages tels que JavaScript.

Pour SPIN (SPARQL Inferencing Notation) [Fürber and Hepp, 2010], il définit une collection légère de classes et de propriétés RDF [12] permettant d'exprimer les requêtes utilisateur et les règles métiers via le langage d'interrogation SPARQL [13] et les exécuter sur la base de connaissances [Knublauch et al., 2011]. Ce formalisme comprend également une bibliothèque qui étend ce langage avec des fonctions communes qui peuvent être réutilisées plusieurs fois.

En ce qui concerne SWRL (Semantic Web Rule Language) [Horrocks et al., 2004], il représente la combinaison de langage d'ontologie OWL DL [Motik et al., 2005] et le langage de règles RuleML [Boley et al., 2010] permettant de mettre en place des règles logiques dans le but de vérifier la cohérence de la base de connaissances ou d'en déduire de nouvelles. SWRL utilise une syntaxe de haut niveau pour modéliser les règles logiques sous forme d'une séquence d'axiomes et de faits. Bien que ce langage ne soit pas recommandé par le W3C, il est largement utilisé dans le domaine du Web sémantique notamment en raison de sa simplicité et de son utilisation intuitive.

1.4 Systèmes Multi-Agents (MAS)

L'aspect dynamique, distribué et ouvert des systèmes IoT rend la résolution des problèmes complexes plus difficile ce qui exige des efforts collectifs entre les différentes entités composant ce genre de système. Le paradigme des systèmes multi-agents (MAS) est une solution pour atteindre cet objectif, car ils permettent d'assurer la décentralisation, l'autonomie, l'adaptabilité et la perception continue de l'environnement pour répondre dynamiquement aux attentes des applications IoT en fonction des contraintes qui leur sont imposées.

12. https://www.w3.org/RDF/
13. https://www.w3.org/TR/rdf-sparql-query/

1.4.1 Agents

Vu l'ancienneté de ce terme, plusieurs définitions ont été proposées.

Ferber et al [Ferber, 1997] définit le terme agent comme suivant : « un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agents, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents ».

Dans [Shoham, 1990], un agent est défini comme « une entité qui fonctionne continuellement et de manière autonome dans un environnement où d'autres processus se déroulent et d'autres agents existent ».

D'un point de vue logicielle, IBM [Gilbert, 1997] définit les agents comme : « des entités logicielles qui réalisent des opérations à la place d'un utilisateur ou d'un autre programme, avec une sorte d'indépendance ou d'autonomie, et pour faire cela, ils utilisent une sorte de connaissance ou de représentation des buts ou des désirs de l'utilisateur ».

1.4.2 Caractéristique d'un agent

Plusieurs caractéristiques peuvent être tirées à partir des différentes définitions données précédemment, notamment :

- Autonomie : représente la capacité d'un agent de raisonner sur ses capacités et son état interne pour agir sans l'intervention d'une entité tierce dans le but d'atteindre ses objectifs.
- Flexibilité : l'agent doit agir sur l'environnement en fonction de ses entrées sensorielles représentées via des capteurs qui se chargent de percevoir son environnement et d'acquérir des connaissances sur ce dernier et via des actionneurs qui lui permettent de réagir à ses changements et d'élaborer des réponses tout en optimisant le temps de réponse.
- Rationalité : ce concept sert à définir et mesurer la manière avec laquelle l'agent réagit et prend ses décisions. Un agent doit se comporter de façon à optimiser ses intérêts et les intérêts de ces collaborateurs.
- Réactivité : représente la capacité de l'agent à répondre rapidement et efficacement aux sollicitations de son environnement ou aux demandes de ses homologues ;
- Proactivité : l'agent doit pouvoir prendre l'initiative dans ses décisions avant que les évènements réels les représentant n'aient lieu.
- Sociabilité : représente l'habilité de l'agent à interagir et coopérer avec les autres agents de son système dans le but de résoudre et d'accomplir des tâches complexes.
- Apprentissage : un agent peut bénéficier de ses expériences antérieures et de ses interactions avec d'autres agents dans ses décisions.

- Mobilité : le code d'un agent peut migrer d'un hôte à un autre (d'une machine à une autre) et éventuellement se dupliquer.

1.4.3 Système multi-agents

Les agents ont des connaissances et des capacités limitées et ne possèdent qu'une vue partielle de leur environnement et par conséquent, ils doivent se coordonner entre eux et échanger leurs informations pour résoudre les problèmes complexes et atteindre leurs objectifs. Dans ce contexte, les systèmes multi-agents émergent comme une solution prometteuse à cet effet. Pour définir ce concept, nous utilisons la définition suivante donnée par [Ferber, 1997] qui regroupe tous les éléments de base qui le composent :

Nous appelons un système multi-agent (ou MAS), un système composé de :

- Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.
- Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est à dire qu'ils peuvent être perçus, crées, détruits et modifiés par les agents.
- Un ensemble A d'agents, qui représentent les entités actives du système.
- Un ensemble de relations R qui unissent les objets et les agents entre eux.
- Un ensemble d'opérations Op permettant aux agents A de percevoir, produire, consommer, transformer et manipuler les objets de O.
- Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers» [Ferber, 1997].

Caractéristiques des MAS

L'intelligence conférée aux agents est matérialisée par leur capacité à résoudre les problèmes complexes via leur coopération ce qui garantit la décentralisation et l'autonomie de processus de décision tout en dépendant des informations échangées entre eux via les mécanismes de communication et de collaboration. Les principales caractéristiques de ces systèmes sont [Quinqueton, 2013] :

- La collaboration : elle représente l'acte de faire travailler deux ou plusieurs agents pour élaborer une solution négociée et résoudre un problème complexe afin d'atteindre leurs objectifs en commun. Elle désigne la répartition des tâches entre eux ainsi que les informations utiles pour leur réalisation et les ressources nécessaires pour leur déploiement. Les relations et les rôles entre les agents collaboratifs ne sont pas définis à priori contrairement aux agents qui se coordonnent en se basant sur des relations bien identifiées pour éviter tout sorte de conflit entre eux.

- La communication [Bensaid et al., 1997] : il s'agit de l'action de transmettre et d'échanger les informations et les connaissances entre les agents pour la mise en œuvre des moyens de collaboration et de coordination pour la résolution des problèmes complexes. Nous distinguons deux types de communication :
  - Communication directe : elle consiste à dialoguer par envoi de messages.
  - Communication indirecte : elle est utilisée généralement par les agents réactifs qui utilisent l'environnement comme un médium pour partager les informations entre eux.
- La négociation : c'est une pratique qui consiste à interagir pour rechercher un accord qui maximise les intérêts individuels ou collectifs des agents. Au cours de ce processus, les agents échangent et énoncent des propositions et les évaluent dans le but d'établir un contrat sur un objectif en commun. Une fois que le contrat est établi et un accord est atteint, ce processus se termine. La négociation peut prendre différentes formes dont les principales sont :
  - Le protocole Contract-Net [Chaib-Draa et al., 2001] : pour établir un contrat entre les agents, un message de découverte est diffusé tout d'abord par l'agent gestionnaire à l'ensemble des agents entrepreneurs de système. Au cours de la découverte, le gestionnaire sollicite des propositions en prenant en compte les pré-conditions nécessaires pour l'exécution de ses tâches et il choisit les partenaires les plus appropriés en fonction de leurs caractéristiques.
  - Le mécanisme d'élection [Jamont and Occello, 2006] : la décision à prendre entre les agents et le nouvel état possible du système est le résultat d'un vote dans lequel chacun des agents exprime son opinion en attribuant une valeur d'utilité au prochain état possible du système ou une valeur de confiance à un agent donné.
  - La négociation soumise à des contraintes [Huget and Koning, 2001] : dans ce modèle, les agents sont dotés de capacités limitées et des ressources restreintes qui dirigent leur comportement et leurs décisions.
  - La négociation appelée à prendre ou à laisser (Take it or Leave it) [Jiang et al., 2006] : cette forme de négociation est réalisée en un seul tour dans lequel une ou plusieurs propositions sont formulées et diffuser à l'ensemble des agents qui peuvent soit les accepter ou les refuser et d'avoir la possibilité de contre-proposer ou de renégocier leurs décisions.

1.4.4 Approches des MAS

Nous pouvons distinguer essentiellement trois catégories d'agents en fonction de leur complexité et la manière avec laquelle ils se comportent [Khezami et al., 2005] :

- Approche cognitive [Daudé, 2005] : dans cette approche, chaque agent est doté d'une représentation symbolique explicite de son environnement, de son état interne ainsi que d'autres agents. Par conséquent, il peut formuler son comportement et ses raisonnements à partir de ces connaissances et anticiper ses décisions et celles de ses agents homologues. Cette approche est adoptée généralement par les systèmes qui se composent d'un petit nombre d'agents qui peuvent s'organiser et apprendre à coopérer en tenant compte de leurs expériences passées toute en prévoyant les résultats futurs de leurs actions. Cependant, cette approche est très difficile à mettre en œuvre.
- Approche réactive [Daudé, 2005][Abras, 2009] : les agents de cette approche ne possèdent aucun modèle de leur environnement et se comportent comme des automates et réagissent par stimulus-réponse aux événements qui leur arrivent. Ces agents sont caractérisés par un comportement très simple, intelligent et rapide dans leur prise de décisions, car ils sont dotés de connaissances limitées et des modèles de raisonnement représentés via des règles simples if-then-else. Contrairement à l'approche cognitive, les sociétés composées de ce genre d'agent sont de grande taille et ne considèrent plus leurs expériences passées dans leurs actions pour atteindre leurs buts. L'inconvénient de cette approche est essentiellement le manque de modèles formels.
- Approche hybride [Müller, 2002] : cette approche vise à surmonter les lacunes des deux approches précédentes en dotant les agents d'une structure modulaire qui assure à la fois la cognition et la réactivité. Cette approche permet d'améliorer les capacités de traitement des agents, car leurs composants peuvent fonctionner simultanément. Elle permet également de garantir l'évolution ainsi que la maintenance du système. L'approche hybride a essentiellement pour objectif d'associer les capacités de réaction rapide des agents réactifs aux stimulus de leur environnement aux capacités de raisonnements et d'intelligence des agents cognitifs.

1.5 Du cloud au fog computing : une vue d'ensemble

Les systèmes IoT sont composés de deux parties incontournables : une partie frontale (frontend) qui comprend les différents objets, et une partie dorsale (backend) composée des systèmes assurant le stockage et le traitement des données issues de ces objets. Afin de faire face aux problèmes liés au manque de ressources au sein des périphériques IoT et de leurs capacités limitées, une panoplie d'infrastructures ont été proposées pour permettre le stockage et le traitement des applications IoT tout en assurant des performances assez élevées et un passage à l'échelle.

1.5.1 Cloud computing

Le cloud computing [JoSEP et al., 2010] est une solution qui est apparue essentiellement pour faire face aux lacunes des serveurs traditionnels en terme de gestion de données, de bande passante et de coût. Initialement, le cloud a été mis en avant dans le domaine de l'e-commerce. Ensuite, ses utilisations sont devenues nombreuses, notamment dans le cadre des fournisseurs de ressource comme [Kim et al., 2016][Jennings, 2010][Bonomi et al., 2012][Zhang et al., 2010] : Azure IoT, IBM Watson, AWS et Google.

Le cloud Computing ou l'informatique en nuage [Zhang et al., 2010] est définie comme étant une infrastructure composée de plusieurs serveurs informatiques et d'un nombre important de ressources physiques, souvent homogènes et capables d'assurer les opérations de stockage et de gestion d'énormes quantités de données issues des différents objets connectés. La virtualisation représente la caractéristique la plus importante et le concept clé de cette technologie qui garantit ses performances en terme d'espace de stockages et de puissance de calcul. L'ensemble des ressources cloud peuvent être déployées dans des localisations géographiques différentes, mais elles sont souvent gérées d'une manière centralisée.

Une définition plus formelle et plus générique est donnée par le NIST. Ils définissent le cloud comme : « le cloud computing est un modèle qui permet d'accéder au réseau, de façon ubiquitaire facile et à la demande, à un ensemble de ressources informatiques partagées et reconfigurables (réseaux, serveurs, stockage, applications et services). Ces ressources peuvent être fournies ou libérées avec un système de gestion minimal et selon les interactions avec le fournisseur de services » [Mell and Grance, 2011].

1.5.2 Fog computing

Malgré les avantages multiples que le cloud computing offre, il représente plusieurs limitations notamment à cause de la prolifération de nombre d'objets connectés et les quantités énormes de données issues de ces dispositifs qui sont souvent contraints par leurs capacités. Ces limites font que certaines applications IoT notamment les applications en temps réel (véhicules connectés) ne pourront pas attendre que les données IoT parviennent au cloud, soient analysées et que les actions soient envoyées en réponse aux actionneurs finaux pour effectuer les actions appropriées. Dans ce contexte et dans le but de faire face aux lacunes illustrées précédemment, le fog computing [Bonomi et al., 2012] est apparue comme une solution très prometteuse notamment dans le domaine de l'IoT.

Le fog computing ou l'informatique en brouillard [Yi et al., 2015b] n'est qu'une extension des capacités de calculs et de stockages des serveurs cloud computing aux périphériques réseaux proches des utilisateurs finaux. Ce paradigme a été inventé par Cisco en 2014 [Cisco, 2014]. Le fog computing est défini par le NIST [Iorga et al., 2017]

comme : « le brouillard informatique est un paradigme de ressources horizontales, physiques ou virtuelles, qui réside entre les terminaux intelligents et les centres de données (data centers) de cloud. Ce paradigme prend en charge les applications isolées verticalement et sensibles au temps de latence en fournissant une connectivité informatique, de stockage et de réseau omniprésent, évolutif, en couches, fédérée et distribuée ».

La définition de NIST illustre que le fog représente une couche médiatrice intelligente entre la couche physique regroupant l'ensemble des dispositifs utilisateurs et la couche cloud qui permet essentiellement d'accélérer les traitements. Les technologies de fog et de cloud computing sont complémentaires et l'utilisation d'une d'entre eux n'écarte pas l'usage de l'autre, mais la complète en rapprochant la délivrance des services dans la mesure du possible au plus proche des sources d'informations. Le fog a la possibilité d'interagir avec la couche Cloud pour assurer les besoins des applications non-sensibles à la latence et pour des fonctionnalités de stockage à long terme. Les nœuds fog peuvent être de type cloudlets et des mini centres de données à petite échelle ce qui rend ces nœuds plus puissants et capables de supporter des applications gourmandes en ressources de type temps réel [Dolui and Datta, 2017]. Un des points forts de cette technologie est qu'elle permet de prendre en charge l'hétérogénéité des différentes ressources et interface de connectivité des appareils utilisateur et la mobilité de ces derniers [Yi et al., 2015a].

1.5.3 Fog Computing Vs Cloud Computing

Bien que les concepts de fog computing et cloud computing présentent plusieurs similarités, ils sont bien différents sur certains points. Ci-dessous, un récapitulatif illustrant les principales différences sous forme d'une comparaison point à point [Yannuzzi et al., 2014] [Jalali et al., 2016] [Vaquero and Rodero-Merino, 2014].

- Architecture : la différence clé entre les technologies cloud et fog réside dans le fait que l'infrastructure cloud est centralisée et est constituée de plusieurs centres de données réparties à travers le monde, tandis que, le fog possède une architecture décentralisée et distribuée constituée des millions de nœuds capables d'effectuer des traitements avancés.
- Latence : les serveurs cloud sont souvent situés à des millions de kilomètres des périphériques utilisateur contrairement aux nœuds Fog qui sont proches des sources d'informations ce qui rend la latence cloud plus importante que celle de fog Computing.
- Réactivité : le temps de réponse des systèmes utilisant l'architecture fog est plus faible par rapport à celui fourni par les serveurs cloud. Cette différence est due essentiellement au fait que les nœuds fog sont très proches de la couche de périphériques.

- Sécurité : dans le cloud computing, le traitement de données s'effectue dans des serveurs distants dont la localisation est inconnue pour les utilisateurs ce qui diminue le niveau de sécurité et expose leurs données privées aux malveillances. La sécurité dans le fog est relativement élevée car les nœuds des traitements sont connus par les utilisateurs ou même leurs appartiennent.
- Puissance : en termes de capacités informatiques, de traitement, de stockage et de ressources physiques, le cloud est plus puissant que le fog computing.
- Répartition géographique : le fog comprend un nombre très important de petits nœuds répartis géographiquement ce qui garantit une prise en charge instantanée des demandes des utilisateurs contrairement au cloud.
- Connectivité : dans l'infrastructure fog, plusieurs canaux d'interconnectivité sont disponibles pour véhiculer les données produites par les différents périphériques et les demandes utilisateurs ce qui diminue les risques de défaillance en terme de connectivité. Contrairement au cloud qui risque de s'effondrer si la connexion Internet est perdue.

Un récapitulatif synthétisant les principales différences entre les technologies fog et cloud est donné dans la Table 1.1.

	Fog Computing	Cloud Computing
Architecture	Distribuée	Centralisée
Latence	Faible	Élevée
Location	Très proche des périphériques utilisateur	Loin des périphériques utilisateur
Réactivité	Instantanée	Longue
Sécurité	Niveau élevé	Niveau bas
Puissance	Limitée	Très importante
Répartition géographique	Dense	Moins dense
Nombre nœuds	Important	Limité
Connectivité	Divers protocoles et standards	Internet

Table 1.1 – Fog Computing Vs Cloud Computing

1.6 Découverte et sélection des services : État de l'art

Dans les sections 1.2, 1.3, 1.4, et 1.5, nous avons introduit les systèmes orientés services, le Web sémantique, les Systèmes multi-agents et le fog/cloud computing intervenant dans le développement des applications distribuées. La mise en œuvre de ce

genre d'applications n'est pas envisageable via un seul composant, mais par plusieurs composants appelés services. Ce processus est nommé composition de services. Ce dernier engendre plusieurs difficultés en terme de localisation et de sélection de services, et d'implémentation de leurs interactions. Cette section est consacrée à l'étude des travaux de la littérature sur la découverte, la sélection et la composition de services afin de tirer profit de leurs avantages et mettre l'accent sur leurs limites dans le but de les combler et pouvoir proposer des solutions.

1.6.1 Découverte des services

La découverte de services est une étape très importante dans le cycle de vie de composition des services. Elle désigne le processus qui permet de localiser les services qui répondent aux besoins de l'application à mettre en place et qui possèdent les capacités et les propriétés souhaitées. Les approches de découverte de services sont principalement réparties en deux classes : les approches centralisées et les approches distribuées.

Approches centralisées

Les approches centralisées se basent généralement sur un registre central UDDI qui permet de regrouper les descriptions de tous les services publiés par les fournisseurs dans le même endroit pour faciliter le processus de découverte. Un UDDI peut être public et accessible à nombre illimité de consommateurs ou privé et dédié à un groupe spécifique d'utilisateurs comme en entreprise. Au début de ses utilisations, ce modèle était limité à des recherches syntaxiques simples avec des mots-clés et ne permettait pas de faire un raisonnement ou une inférence pour faire correspondre une requête à une description d'un service. Le travail de [Zhou et al., 2009] fournit une méthode d'indexation inverse du registre de services central pour assurer la découverte rapide des services. Il repose sur l'utilisation d'un ensemble de mots-clés liés aux identifiants des documents dans lesquels se trouvent ces mots-clés. Cependant, ce processus d'indexation a besoin d'espace de stockage supplémentaire. Les auteurs de [Grigori et al., 2010] proposent un modèle basé sur la transformation de processus BPEL (Business Process Execution Language) qui exprime les besoins des utilisateurs dans un graphe de comportement. La transformation est effectuée en appliquant une technique d'appariement comportemental. Par conséquent, le problème de découverte est mappé à un problème de correspondance de graphiques. Une distance de similitude est calculée entre le graphe des services demandés et celui des services fournis. Les services avec une petite distance sont retournés. Cependant, cette méthode a un coût de calcul élevé et n'utilise que la correspondance syntaxique. En plus, les approches syntaxiques dans leur globalité présentent des insuffisances en termes de manque d'exactitude des résultats.

Pour faire face aux limites des méthodes syntaxiques, une panoplie de modèles sémantiques ont été proposés [Batra and Bawa, 2011][Pukkasenung et al., 2010]

[Lu et al., 2012b]. Dans le travail de [Srinivasan et al., 2004], les auteurs proposent d'utiliser un nouveau composant appelé OWL-S Matchmaker basé sur raisonneur Racer 4, qui permet de découvrir les services requis en utilisant un appariement sémantique entre la requête utilisateur et les descriptions des services disponibles. Dans [Kourtesis et al., 2008], les auteurs proposent une solution qui adopte SAWSDL (Semantic Annotations for WSDL) comme modèle d'annotation sémantique des interfaces des services Web et OWL-DL (OWL- Description Logic) pour modéliser leurs capacités ce qui permet une découverte plus précise des services Web. Un modèle sémantique de publication/abonnement pour la découverte de services pour les systèmes IoT est proposé dans [Dong and Chen, 2016]. Dans ce modèle, les utilisateurs souscrivent aux sujets de leurs intérêts dans le registre UDDI et lorsqu'un nouveau service correspondant à leurs attentes est publié dans ce registre, une notification est envoyée aux utilisateurs abonnés.

Les approches centralisées basée sur un UDDI qu'elles soient syntaxiques ou sémantiques étaient auparavant largement adoptées. Néanmoins, avec l'accroissement exponentiel du nombre des services Web disponibles sur le web, et l'augmentation du nombre d'utilisateurs à l'échelle mondiale, plusieurs problèmes critiques peuvent être soulevés. Tout d'abord, ces approches reposent sur une seule entité coordinatrice qui se charge de la gestion de l'ensemble du système ce qui les rendent très limitées et en cas de panne, le système devient défaillant et non fonctionnel. En plus, elles sont très coûteuses parce qu'elles nécessitent d'importantes ressources et imposent des contraintes logicielles et matérielles pour le stockage, la gestion et la maintenance de l'ensemble des services face au nombre important des requêtes requises.

Approches distribuées

Pour faire face aux problèmes d'évolution et de disponibilité des approches centralisées traditionnelles, les approches distribuées basées sur les systèmes multi-agents sont proposées. Elles visent à automatiser la découverte des services et limiter les interactions avec les utilisateurs [Sycara et al., 2004][Cao et al., 2001].

Dans [Palathingal et al., 2004], les auteurs proposent une nouvelle approche basée sur les agents appelée AASDU (Agent Approach for Service Discovery and Utilization). Elle utilise des agents capables d'interagir avec les utilisateurs finaux et qui ont pour rôle d'accepter leurs requêtes, découvrir les services requis et gérer efficacement leur invocation tout en permettant la communication et la coopération avec d'autres agents. [Kang and Sim, 2011] proposent un système de Courtage de services (service brokering) qui permet d'utiliser des agents courtiers (brokers) pour la découverte des services au niveau de Cloud. Dans ce système, les différents utilisateurs (consommateurs ou fournisseurs) sont représentés via des agents qui se chargent d'envoyer leurs

requêtes (demandes ou publications) à un agent courtier et d'afficher les résultats. Les agents courtiers ont pour rôle de réaliser les opérations de découverte et de recommandation en se basant sur les données historiques d'autres agents de courtier de la base de données. Si les files d'attente de cet agent sont surchargées, il renvoie les requêtes à d'autres agents courtiers pour équilibrer les charges de travail. Un autre travail intéressant est celui de [Muller et al., 2006] basé sur le concept d'agent-service. Le but de la composition à satisfaire (workflow générique) est publié par un agent utilisateur dans un tableau noir. Après l'enregistrement, chaque agent de service récupère les sous-objectifs du tableau noir pour les évaluer et savoir s'ils peuvent contribuer à leur réalisation. [Marengereke and Krishnan, 2015] ont proposé une plateforme appelée Eksarva qui permet de modéliser formellement le processus de collaboration comme un flux de travail (workflow) à base de règles logiques pour assurer que le travail de groupe soit pertinent. Dans l'approche proposée par [Sellami et al., 2016], les agents sont dotés de ressources restreintes et sont soumis à des contraintes temporelles fortes et chacun d'entre eux vise à améliorer progressivement les délais de réponse. Pour cela, les agents effectuent plusieurs échanges afin de formuler une coalition qui maximise l'utilité de collaboration tout en minimisant le nombre de messages échangés et les temps de négociation et de réponse. Dans le travail de [Chareton et al., 2017], les auteurs ont proposé un nouveau langage, nommé KHI, dont la sémantique est décrite par la logique USL (Updatable Strategy Logic) où les capacités et les objectifs des différents agents du système sont formulés via la logique temporelle linéaire. Chacun des objectifs est assigné à une coalition d'agents et dans leur travail, ils cherchent à corriger et affiner ces assignations en les évaluant en terme de pertinence via plusieurs échanges et négociations entre les agents. Pour cela, ils ont présenté une procédure décidable de vérification qui a pour but de transformer le problème de satisfaction des critères en un problème de vérification du modèle (model-checking) en tenant compte des capacités des agents.

Contrairement aux approches traditionnelles basées sur le mode descendant pour la composition des services, le travail de [Younes et al., 2018] propose une nouvelle approche ascendante où les services sont découverts et composés à la volée à partir des services disponibles au moment de l'exécution dans des environnements IoT sans se baser sur une description préétablie de l'application voulue. L'architecture de leur approche comprend : a) un ensemble d'agents où chaque agent est attaché à un service qu'il soit fourni ou requis, b) une entité appelée Probe qui se charge de surveiller l'environnement et détecter les apparitions et disparitions des agents, c) une entité appelée médiateur (mediator) qui assure la communication entre les agents afin de trouver des connexions appropriées avec un autre service présent dans l'environnement et d) une derrière entité appelée fabrique (factory) qui permet de créer des compositions appropriées. Cependant, ce mode descendant de découverte et composition de services IoT n'est pas très approprié parce que les compositions créées à la volée peuvent ne correspondre à aucun des besoins utilisateur.

Dans [Berrani et al., 2018], une approche multi-agents pour composition des services IoT est proposée. Elle est conçue en utilisant le langage SysML et implémentée via la plateforme Netlogo. Dans cette proposition, plusieurs agents sont engagés à découvrir les services IoT qui peuvent satisfaire une demande utilisateur.

Les principaux inconvénients des méthodes de découverte basée sur MAS résident dans la complexité et la difficulté de leur mise en œuvre en termes d'implémentation d'agents, leur communication ainsi que leur maintenance. En plus, vu leur complexité, elles sont généralement utilisées dans des environnements fermés et limités comme l'entreprise pour des échanges B2B (Business to Business) par exemple. Très peu de travaux basés sur les MAS se sont intéressés au problème de découverte des services IoT. Dans ces contributions, la problématique qui se pose concerne l'hétérogénéité des objets IoT et la diversité des architectures, des formats de données et des protocoles utilisés. En effet les agents représentant les objets IoT ne sont pas basés sur un standard dans leur modélisation et chaque travail peut concevoir sa propre modélisation ce qui pose le problème de fragmentation du marché IoT causé par la diversité des normes et des modèles utilisés.

1.6.2 Sélection des services

Une fois que la phase de découverte est achevée, plusieurs services candidats possédant des fonctionnalités similaires mais des propriétés non-fonctionnelles différentes notamment en terme de QoS, seront disponibles. La phase de sélection [Lopez-Velasco, 2008] consiste à choisir les services les plus adéquats pour les différentes tâches à satisfaire, parmi ceux qui ont été précédemment découvert en se basant sur leurs propriétés non-fonctionnelles (QoS). Le problème de sélection de services a attiré beaucoup d'attention dans la littérature dans de multiples domaines scientifiques et industriels. Les solutions proposées peuvent être classées principalement en deux catégories : les solutions basées sur la sélection globale et celles basées sur la sélection locale.

Sélection globale

La sélection globale [Fan et al., 2015] [Qiqing et al., 2009] [Liu and Xu, 2014] a pour objectif de choisir la combinaison optimale via des algorithmes exhaustifs, ou quasi-optimale en appliquant des méthodes heuristiques, parmi toutes les compositions possibles tout en respectant les contraintes de QoS de bout en bout des utilisateurs et leurs préférences. Dans [Liu et al., 2017], les auteurs utilisent la méthode MILP (Mixed-integer linear programming) pour sélectionner efficacement les services SaaS qui peuvent répondre aux exigences de QoS des utilisateurs. Cette méthode utilise le calcul Skyline pour déterminer les services qui matchent aux mieux avec les demandes utili-

sateur dans la bibliothèque de services SaaS. Rajeswari et al. [Rajeswari et al., 2014] ont adopté une méthode de planification globale pour trouver toutes les combinaisons de services possibles pour répondre à un besoin et sélectionner la composition avec le score le plus élevé. Le travail de [Yu et al., 2007] utilise un algorithme appelé WS-IP pour modéliser le problème de sélection basée sur la QoS comme un problème IP 0-1 et utilise la méthode de séparation et d'évaluation progressive (branch and bound) pour trouver la composition de services optimale. L'inconvénient majeur de ces méthodes à force brute est qu'elles sont NP-difficiles et leur complexité temporelle est d'ordre exponentiel, ce qui les rend non évolutives.

Pour faire face à ces lacunes, plusieurs travaux basés sur des méthodes approximatives utilisant des algorithmes méta-heuristiques qui visent à trouer des compositions quasi-optimales sont proposés. Dans [Chifu et al., 2017], un algorithme hybride des colonies d'abeilles (hybride Honey Bees Algorithm, hHBA) pour sélectionner la composition quasi-optimale des services Web sémantiques est proposé. Un algorithme amélioré d'optimisation par essaim de particules (improved Particle Swarm Optimization Algorithm, iPSOA) est fourni dans [Wang et al., 2010] pour résoudre le problème de sélection basé sur la QoS. Il propose plusieurs améliorations comme l'application d'une stratégie de mutation non-uniforme (Non-Uniform Mutation, NUM) à la meilleure particule globale pour améliorer la diversité de la population et surmonter la prématurité de la méthode PSOA standard, les stratégies d'ajustement adaptatif du poids (Adaptive Weight Adjustment, AWA) et la stratégie de meilleur local d'abord (Local Best First, LBF) pour améliorer la vitesse de convergence de l'algorithme. Dans [Karimi et al., 2017], les auteurs proposent un algorithme génétique (Genetic Algorithm, GA) pour la sélection globale des services dans le Cloud Computing. Cet algorithme est combiné à des techniques de clustering permettant de faciliter l'exploration de données et réduire l'espace de recherche.

Bien que les travaux basés sur des algorithmes méta-heuristiques atteignent de bonnes performances par rapport aux méthodes exhaustives, ils ne sont pas souhaitables pour les environnements IoT distribués, car ils reposent sur un gestionnaire central ce qu'il les rend limités par le problème du point d'échec unique. En plus, le parcours de toutes les compositions possibles est très coûteuse.

Sélection locale

La sélection locale [Alrifai et al., 2010] [Qi et al., 2010] [Guidara et al., 2015] est une stratégie qui consiste à sélectionner le service le plus approprié pour chaque tâche abstraite indépendamment des autres tâches composant le processus à accomplir en explorant les contraintes de QoS relatives à la tâche considérée tout en ignorant les critères de QoS relatifs au processus dans sa globalité. Pour parvenir à considérer les exigences de QoS globales, la technique de décomposition des contraintes de QoS

globales en contraintes locales est intégrée. Ces contraintes locales sont utilisées comme bornes pour la sélection locale parallèle pour chaque tâche.

Alrifai et al. [Alrifai et al., 2012] ont proposé une approche intéressante et assez complète pour décomposer les contraintes de QoS globales en considérant les structures séquentielles, en boucle, parallèles et conditionnelles. Leur méthode commence par diviser la plage de valeurs de chaque paramètre de QoS en plusieurs valeurs discrètes (niveaux de qualité) qui seront utilisées comme contraintes locales candidates. Ensuite, ils utilisent la technique MILP pour résoudre le problème d'optimisation de la recherche de contraintes de QoS locales. Cependant, cette méthode n'est pas évolutive car elle obtient de mauvaises performances lorsque le nombre de services candidats et les contraintes de QoS sont relativement importants. Une nouvelle méthode heuristique qui divise le problème de décomposition en plusieurs sous-problèmes est proposée dans [Yanwei et al., 2010]. Elle se réalise en deux étapes : a) la première étape consiste à déterminer une solution initiale qui répond à toutes les contraintes globales, b) la deuxième étape, quant à elle, consiste à mettre à jour progressivement cette solution jusqu'au trouver une solution quasi-optimale. Une autre méta-heuristique est proposée dans [Yuan et al., 2019]. Elle est basée sur la technologie de la logique floue et l'algorithme génétique culturel (CGA) pour la décomposition des contraintes de QoS globale en contraintes locales. Cette méthode vise à trouver les contraintes locales qui permettent de sélectionner la composition de services quasi-optimale pendant l'exécution.

Bien que les travaux proposés pour la décomposition des contraintes de QoS globale en contraintes locales atteignent de bonnes performances, aucune d'entre eux n'a couvert la fiabilité de la composition résultante. Ils ignorent que les valeurs des propriétés de QoS sont dynamiques, et que les valeurs des attributs QoS fluctuent.

1.6.3 Composition de services

La composition de services représente l'action de former un service composite à valeur ajoutée par l'association des fonctionnalités de plusieurs services qui peuvent être à leur tour atomiques ou composites. La mission achevée par le service composite ne peut être accomplie par un seul service.

Un nombre important de travaux ont été menés dans cette thématique ce qui a donné naissance à plusieurs orientations et approches de composition. Ces approches peuvent être classées en approches statiques et approches dynamiques. Selon [Fki, 2015], ces deux approches se distinguent par rapport au temps d'intégration des services concrets sélectionnés dans la composition. Les approches statiques optent pour l'intégration des services au moment de la conception tandis que les approches dynamiques élisent les services au cours de la phase d'exécution.

Approches de composition statiques

Elles sont définies à l'aide d'un processus métier (business process) et elles sont à leur tour divisés en deux catégories : orchestration et chorégraphie.

Pour l'approche d'orchestration de services [Amato et al., 2016] [Wen et al., 2017] [Gortmaker et al., 2004], le processus de coordination et d'enchaînement des services pré-sélectionnés dans la phase de sélection est fait dans un ordre bien spécifique selon des scripts d'orchestrations fournis en avance. Ces scripts permettent de modéliser les différentes relations et interactions entre les services sélectionnés ainsi que les branchements logiques existants entre eux et la séquence de leur invocation lors de l'exécution. L'orchestration est menée et contrôlée par un coordinateur central (orchestrateur). Il existe plusieurs standards de modélisation de processus d'orchestration. Parmi eux, on trouve par exemple : BPMN (Business Process Modeling Notation) [Allweyer, 2016] et BPEL (Business Process Execution Language) [Bennett et al., 2016]. Cette approche est limitée par son coté centralisée et statique. Par conséquent, en cas d'échec ou de modification des besoins de l'application, la composition devient invalide.

Concernant la méthode de chorégraphie de services [Charif, 2007] [Sajadi et al., 2016], elle permet de décrire les interactions entre un ensemble de services de manière collaborative en modélisant les séquences des messages échangés entre les différentes sources impliquées notamment les fournisseurs et les clients. Dans ces approches, chaque service participant au processus de composition décrit son rôle et connaît les services avec lesquels il peut interagir de manière autonome, sans l'intermédiaire d'un autre composant. Le standard WS-CDL (Web Services Choreography Description Language) [Ebrahimifard et al., 2016] représente un des standards de chorégraphie les plus utilisés. Les limites de cette approche résident dans le fait que si les besoins ou les partenaires de chorégraphie changent, les collaborations deviennent impossibles. De plus, il n'existe pas de langage permettant d'exprimer d'une façon dynamique les besoins de l'application.

Approches de composition dynamiques

Elles concernent les techniques de composition à la volée qui sont capables de découvrir les services disponibles à un instant donné ainsi que leurs fonctionnalités pour satisfaire un besoin donné et adapter la composition en fonction de contexte de l'environnement d'exécution. Dans la suite, nous décrivons les principales approches incluses dans la catégorie de composition dynamique à savoir : les techniques orientées intelligence artificielle, les techniques à base de règles et les techniques basées sur le chaînage.

Pour les techniques orientées intelligence artificielle, une très grande variété de travaux de recherche a exploité les progrès de la planification IA et les a intégrés dans les techniques de composition de services [Guitton and Fiorino, 2006] [Peer, 2004] [Guitton and Fiorino, 2006] [Sheshagiri et al., 2003] [El Falou, 2010] [Seghir, 2018]. La

planification basée sur le réseau hiérarchique des tâches (HTN pour Hierarchical Task Network) [Erol et al., 1995] est une technique qui a prouvé son efficacité dans le domaine de la composition de services. Elle est basée sur le principe de décomposition itérative de l'objectif de composition en sous-tâches atomiques et primitives qui peuvent être accomplies par des opérations élémentaires [Sirin and Parsia, 2004]. Parmi les principaux travaux qui ont adopté cette technique, le travail [Wu et al., 2003]. Dans ce dernier, les auteurs ont proposé un planificateur HTN appelé SHOP2 (Simple Hierarchical Ordered Planner 2) qui se sert des descriptions sémantiques des services pour assurer la composition automatique. [Traverso and Pistore, 2004] proposent de construire le plan de planification via la réalisation de systèmes de transition d'état décrivant les interactions dynamiques des agents. Dans [Durfee, 2001], un agent de gestion est utilisé pour décomposer un objectif en sous-tâches et les attribuer à des agents capables de les réaliser. [Cheng et al., 2002] ont proposé un nouveau langage ASCL (Agent Service Composition Language) qui se base sur un langage de description ASDL (Agent Service Description Language) pour décrire le comportement des services Web et leurs protocoles d'interaction. Il permet également de définir la logique suivie pour la composition des services et l'adaptation de cette composition en fonction de contexte d'exécution. Un autre travail dans ce contexte est fourni par [McDermott, 2002] dans lequel ils utilisent la planification par régression estimée (Estimated-Regression Planning) basée sur les heuristiques pour permettre une recherche optimale des solutions qui sont modélisées par des services composites. Dans les travaux de [Ermolayev et al., 2003] les besoins sont exprimés sous forme de contraintes et de préférences. Les différentes tâches des services sont effectuées par des agents médiateurs, par conséquent, la composition dynamique des services est remplacée par une coalition d'agents.

Quant aux techniques à base de règles, elles sont basées sur un ensemble de règles d'inférence couplées à un ensemble de contraintes et de conditions d'activation. Ces assertions utilisent un raisonnement déductif de type si-sinon-alors (if-else-then) qui représentent la base de connaissances qui détermine le comportement à adopter et guide le processus de composition de services automatique. [Medjahed, 2004] proposent un modèle de composabilité basé sur des règles sémantiques pour vérifier si les services Web peuvent être combinés ensemble tout en évitant les défaillances inattendues au moment de l'exécution. Le modèle proposé se base sur le langage de description de haut niveau CSSL (Composite Service Specification Language) pour décrire les différents services afin de faciliter la phase de correspondance (matching). Un système de composition de service à base de règles a été proposé dans [Mokhtar et al., 2005]. Ce système permet d'étendre les spécifications OWL en intégrant des règles sémantiques qui spécifient la manière dont les applications sensibles au contexte seront composées. Dans le travail de [Chun et al., 2005], les auteurs ont proposé des règles de compatibilité syntaxiques et sémantiques qui intègrent les politiques imposées par les différentes parties de composition (fournisseurs, utilisateurs, etc.) pour permettre une composition

automatique de services. Un autre travail très intéressant est proposé dans le cadre de ces approches est celui de [Zeng et al., 2008]. Dans ce dernier, Zeng et al. proposent une méthode de composition dynamique basée sur des règles métier d'inférence enrichies avec les informations contextuelles.

En ce qui concerne les techniques basées sur le chaînage, elles s'appuient sur des moteurs de raisonnements déductifs capables de traiter et d'apercevoir les dépendances existantes entre les divers services dans le but d'établir un plan de composition qui répond à un objectif donné. Dans ce contexte, [Aversano et al., 2004] proposent d'utiliser un algorithme de chaînage en arrière pour établir une composition de service dynamique en explorant un registre UDDI. L'inconvénient de cette méthode est que le temps de réponse peut être très élevé si le registre contient beaucoup de services indépendants. Le travail de [Mohr et al., 2015] s'appuie également sur le chaînage en arrière pour explorer toutes les compositions de services possibles dans le but de trouver la meilleure. Les algorithmes de chaînage en avant ont été aussi exploités dans la composition dynamique de services, comme le travail de [Arpinar et al., 2005]. Dans ce dernier, ils utilisent l'algorithme de plus court chemin dans le graphe de service via deux passes : la première passe vise à construire les différents plans de composition possibles qui satisfont les sorties (outputs) de la requête et la deuxième permet de rechercher le chemin optimal parmi les compositions construites durant la première passe. [Chan and Lyu, 2008] ont proposé d'utiliser les réseaux de Petri en se basant sur des descriptions WSDL pour composer dynamiquement les services.

1.6.4 Synthèse

Le Tableau 1.2 donne un récapitulatif des travaux orientés services étudiés dans ce travail et cités dans cette section. Une comparaison entre eux est effectuée selon certaines caractéristiques fondamentales et nécessaires pour répondre aux attentes de développement des applications IoT distribuées. Ces caractéristiques sont les suivantes :

- Approche de découverte : pour désigner la méthode utilisée pour localiser les services nécessaires à la composition pour répondre à un besoin applicatif.
- Approche de sélection : pour déterminer l'approche utilisée pour choisir et sélectionner, parmi les services découverts, les services à intégrer concrètement dans la composition de services finale.
- Approche de composition : cette caractéristique permet de souligner la manière avec laquelle les services sélectionnés vont être inter-connectés et composés d'une manière statique à l'avance ou dynamique au cours de l'exécution.
- Méthode : pour désigner la technique ou l'algorithme proposé pour la découverte, la sélection ou la composition de services.
- Architecture : décrit l'architecture et la portée de l'environnement où sera appliqué l'approche proposée par l'outil. Elle est soit centralisée ou distribuée. Il

s'agit d'un critère très important, car il a une relation directe avec la manière de déployer l'application composée et de l'utiliser.

- Description : pour décrire la manière avec laquelle les services sont modélisés. Deux types de description sont possibles : syntaxique ou sémantique.
- Interopérabilité IoT : pour indiquer si le travail étudié s'est basé sur un standard sémantique pour modéliser les services fournis par les objets IoT hétérogènes s'il est appliqué dans le domaine de l'IoT.
- Fluctuation de QoS : pour désigner si le travail de sélection de services étudié considère la nature dynamique des attributs QoS, i.e., la fluctuation des attributs QoS.

Référence	Approche de découverte	Approche de sélection	Approche de composition	Méthode	Architecture	Description	Interopérabilité IoT	Fluctuation QoS
[Zhou et al., 2009]	UDDI	-	-	mots-clés	Centralisée	Syntaxique	✗	✗
[Grigori et al., 2010]	UDDI	-	-	Transformation des graphes	Centralisée	Syntaxique	✗	✗
[Srinivasan et al., 2004]	UDDI	-	-	OWL-S Matchmaker	Centralisée	Sémantique	✗	✗
[Kourtesis et al., 2008]	UDDI	-	-	SAWSDL	Centralisée	Sémantique	✗	✗
[Dong and Chen, 2016]	UDDI	-	-	publication/ abonnement	Centralisée	Sémantique	✗	✗
[Palathingal et al., 2004]	MAS	-	-	AASDU	Distribuée	Syntaxique	✗	✗
[Kang and Sim, 2011]	MAS	-	-	Service Brokering	Distribuée	Syntaxique	✗	✗
[Muller et al., 2006]	MAS	-	-	Tableau noir de publication	Distribuée	Syntaxique	✗	✗
[Marengereke and Krishnan, 2015]	MAS	-	-	plateforme Eksarva	Distribuée	Syntaxique	✗	✗
[Sellami et al., 2016]	MAS	-	-	Coalition d'agents	Distribuée	Syntaxique	✗	✗
[Chareton et al., 2017]	MAS	-	-	Langage KHI	Distribuée	Syntaxique	✗	✗
[Younes et al., 2018]	MAS	-	-	Mode ascendant	Centralisée	Syntaxique	✗	✗
[Berrani et al., 2018]	MAS	-	-	langage SysML, plateforme Netlogo	Distribuée	Syntaxique	✗	✗
[Liu et al., 2017]	-	Globale	-	MILP	Centralisée	Syntaxique	✗	✗
[Rajeswari et al., 2014]	-	Globale	-	Planification globale	Centralisée	Syntaxique	✗	✗
[Yu et al., 2007]	-	Globale	-	WS-IP	Centralisée	Syntaxique	✗	✗
[Chifu et al., 2017]	-	Globale	-	hHBA	Centralisée	Syntaxique	✗	✗
[Wang et al., 2010]	-	Globale	-	iPSOA	Centralisée	Syntaxique	✗	✗
[Karimi et al., 2017]	-	Globale	-	GA	Centralisée	Syntaxique	✗	✗
[Alrifai et al., 2012]	-	Locale	-	MILP	Centralisée	Syntaxique	✗	✗
[Yanwei et al., 2010]	-	Locale	-	Nouvelle heuristique	Centralisée	Syntaxique	✗	✗
[Yuan et al., 2019]	-	Locale	-	CGA	Centralisée	Syntaxique	✗	✗
[Allweyer, 2016]	-	-	Orchestration	BPMN	Centralisée	Syntaxique	✗	✗
[Bennett et al., 2016]	-	-	Orchestration	BPEL	Centralisée	Syntaxique	✗	✗
[Ebrahimifard et al., 2016]	-	-	Chorégraphie	WS-CDL	Centralisée	Syntaxique	✗	✗
[Wu et al., 2003]	-	-	planification HTN	SHOP2	Distribuée	Syntaxique	✗	✗
[Durfee, 2001]	-	-	planification HTN	Agent gestionnaire	Distribuée	Syntaxique	✗	✗
[Cheng et al., 2002]	-	-	planification HTN	Langage ASCL	Distribuée	Syntaxique	✗	✗
[McDermott, 2002]	-	-	planification HTN	Régression estimée	Distribuée	Syntaxique	✗	✗
[Medjahed, 2004]	-	-	A base de règles	Langage CSSL	Distribuée	Syntaxique	✗	✗
[Mokhtar et al., 2005]	-	-	A base de règles	Langage OWL	Distribuée	Sémantique	✗	✗
[Zeng et al., 2008]	-	-	A base de règles	Inférence Contextuelle	Distribuée	Sémantique	✗	✗
[Aversano et al., 2004]	-	-	A base de chaînage	Raisonnements déductif	Distribuée	Syntaxique	✗	✗
[Mohr et al., 2015]	-	-	A base de chaînage	Chaînage en arrière	Distribuée	Syntaxique	✗	✗
[Arpinar et al., 2005]	-	-	A base de chaînage	Algorithme de plus court chemin	Distribuée	Syntaxique	✗	✗
[Chan and Lyu, 2008]	-	-	A base de chaînage	Réseaux de Petri	Distribuée	Syntaxique	✗	✗

TABLE 1.2 – Une synthèse des travaux orientés services

À travers cette étude, nous avons conclu que les méthodes traditionnelles de découverte des services sont généralement centralisées et basées sur un registre UDDI qui regroupe l'ensemble des toutes les descriptions syntaxiques ou sémantiques des services. Cependant, avec l'accroissement exponentiel du nombre des services, ces approches ne sont plus adéquates, car elles exigent d'importantes ressources logicielles et matérielles ce qui les rendent très coûteuses. En plus, elles souffrent du problème du point de défaillance unique. Pour faire face à ces problèmes, les approches MAS sont introduites pour garantir l'extensibilité du système, sa robustesse, et la tolérance aux pannes. Cependant, elles sont très complexes et difficiles à mettre en œuvre et peu d'entre elles se sont focalisées sur la découverte des services dans le domaine de l'IoT. En outre, elles ne gèrent pas le problème de la fragmentation verticale du marché IoT car chacune d'elles propose sa propre modélisation pour garantir l'interopérabilité IoT. C'est pour cela que nous proposons dans ce travail de thèse une nouvelle approche de découverte des services IoT en tirant parti des avantages de navigabilité et de scalabilité fournis par les réseaux sociaux. Notre approche s'appuie sur le WoT pour virtualiser des objets IoT hétérogènes en utilisant une représentation uniforme, standard et sémantique appelée *Avatar*. Un avatar est doté d'un raisonnement autonome et des capacités de collaboration et peut être déployé dans une infrastructure fog/cloud computing.

Concernant les travaux de sélection, deux catégories de travaux sont proposées. Les travaux basés sur la sélection globale et ceux basés sur la sélection locale. Les travaux de sélection globale considèrent toutes les combinaisons de services possibles pour choisir la combinaison optimale ou quasi-optimale via des algorithmes exhaustifs ou des méta-heuristiques respectivement. Par conséquent, ces approches sont très coûteuses en temps et en espace et ne sont plus adaptés aux systèmes distribués à large échelle. Les travaux de sélection locale quant à eux, sont basés sur la décomposition des contraintes de QoS globales en contraintes locales qui servent comme bornes supérieures/inférieures pour sélectionner le service le plus approprié pour chaque tâche composant le processus indépendamment des autres tâches sans avoir à vérifier toutes les combinaisons possibles ce qui permet de remédier aux limites de la sélection globale. Néanmoins, ces approches ne prêtent aucune attention à la fluctuation de QoS qui reflète la variation et la distribution des valeurs d'un paramètre de QoS donné au fil du temps. Il s'agit d'un facteur très important qui garantit la fiabilité de la composition au moment de l'exécution car les services avec des paramètres de QoS très variables peuvent diverger. À cet égard, nous proposons une nouvelle approche de sélection basée sur la décomposition des contraintes de QoS globales en contraintes locales tout en considérant le critère de fluctuation en utilisant un algorithme génétique évolutionnaire et la méthode de machine à vecteurs de support.

Pour les méthodes de composition, nous les avons étudiées afin de trouver la méthode la plus adéquate pour réaliser la composition des services retournés par notre approche de sélection. Comme résultats, nous avons trouvé que les protocoles d'échange

utilisés par les MAS pour construite le plan d'exécution semblent être les plus adéquats.

1.7 Conclusion

À travers ce chapitre, nous avons fourni un aperçu général sur le contexte scientifique et les différents concepts utilisés dans ce travail de thèse ainsi qu'un état de l'art sur la découverte et la sélection des services.

Dans la première partie de ce chapitre, nous avons présenté les notions fondamentales de l'IoT et le WoT et les principaux concepts qui se trouvent au sommet de ces domaines. Nous avons également étudié le paradigme des systèmes orientés services, le Web sémantique et les ontologies des domaines de l'IoT et des services Web, les notions préliminaires liées aux systèmes multi-agents et les technologies fog/cloud computing nécessaires au déploiement à large échelle des systèmes IoT. Dans la deuxième partie, une étude de l'état de l'art et les principales approches existantes dans la littérature pour la découverte et la sélection des services est présentée. Nous avons d'un côté considéré les approches relatives aux services Web traditionnelles qui sont généralement centralisées. De l'autre côté, les approches dynamiques et distribuées réalisées via les systèmes multi-agents sont également étudiées. Une comparaison selon plusieurs critères fonctionnels est menée entre les approches étudiées afin de soulever les failles et les défis à surmonter pour garantir le développement d'applications IoT évolutives.

L'hétérogénéité et la non-standardisation des descriptions des objets IoT rendent la tâche de développement des applications IoT fastidieuse. Par conséquent, il est important de définir un modèle de description standardisée en se basant sur des ontologies afin de pouvoir automatiser le processus de développement IoT. Ce modèle doit être enrichi par des mécanismes de raisonnement autonome afin de permettre aux objets IoT de prendre des décisions intelligentes en fonction de leur contexte et de collaborer avec les autres objets pour accomplir des tâches complexes qu'il ne peut pas réaliser tout seul. Le chapitre suivant sera donc consacré à décrire notre première contribution qui consiste en une architecture modulaire basée sur des artefacts virtuels appelés Avatars qui assurent à la fois l'interopérabilité et l'autonomie des objets IoT hétérogènes pour promettre un comportement collaboratif et intelligent pour l'achèvement des objectifs complexes.

Chapitre 2

Vers une architecture IoT distribuée basée sur des avatars autonomes

Sommaire

Les objets IoT sont de nature hétérogène et déployés dans des environnements hautement distribués et incertains, subissant ainsi des modifications imprévisibles. De plus, ils sont souvent dotés de ressources limitées qui impactent considérablement leurs performances. Le WoT est un pas en avant dans l'IoT qui permet de faire face à ces problèmes. Il s'appuie sur le Web pour offrir une représentation de haut niveau des objets IoT hétérogènes d'une manière uniforme et standardisée. Cela permet de garantir l'interopérabilité et la flexibilité entre les systèmes IoT.

Dans ce chapitre, nous présentons notre première contribution qui consiste à transformer les représentations virtuelles passives proposées dans les approches traditionnelles en entités actives appelées *avatars autonomes*. À travers ces avatars, les objets IoT deviennent actifs et capables de découvrir et de sélectionner d'autres objets afin de collaborer avec eux dans la réalisation d'objectifs complexes en prenant en compte le contexte dans lequel ils se trouvent.

Ce chapitre est organisé comme suit. Tout d'abord, les travaux adoptant la virtualisation des objets sont étudiés. Ensuite, la notion d'avatar est détaillée à travers sa

définition, ses caractéristiques ainsi que ses composants architecturaux. Enfin, l'architecture générique de l'IoT basée sur les avatars et le fog computing est présentée.

2.1 Virtualisation des objets IoT : un panel d'approches

De nombreux travaux liés au concept de virtualisation des objets connectés ont été proposés au cours de ces dernières années à travers des activités de recherche et de développement.

Le travail de [Römer et al., 2004] fait référence aux premières réflexions autour de l'utilisation de la virtualisation pour représenter des objets du monde physique dans le but de les utiliser pour la réalisation des applications informatiques omniprésentes. Ils utilisent le concept de méta-contrepartie virtuelle (Virtual Meta Counterparts, VMC) pour représenter des objets non connectés étiquetés par des étiquettes RFID. Un VMC peut correspondre à un ensemble d'objets similaires appartenant à la même catégorie, permettant ainsi, de simplifier la gestion des différents objets physiques représentés. Ce travail est limité par rapport aux représentations syntaxiques et passives des services fournis par les objets ce qui n'est plus adapté aux besoins des applications IoT notamment en termes de découverte et de sélection des services.

Dans [Han et al., 2013], un système d'automatisation des bâtiments intelligents, où les appareils sont représentés par le concept de profil de périphérique pour le service Web (Device Profile for Web Service, DPWS), est proposé. Ce travail est basé sur le paradigme architectural SOA pour réaliser une composition de services dynamique en fonction des informations contextuelles collectées et traitées. Dans cette architecture, tous les DPWS sont orchestrés et gérés par un serveur central appelé BApS (Building Application Server). Les DPWSs sont passifs et ils n'ont aucun pouvoir de prise de décision dans la composition de services construite.

Dans le travail de [Shamszaman et al., 2014], les auteurs ont proposé un système de gestion des incendies d'urgence en se basant sur une infrastructure WoT. Cette infrastructure permet de mettre à disposition les différents objets intelligents dans le développement des applications IoT et les services à valeur ajoutée en utilisant des représentations virtuelles sur le Web. L'élément clé de ce travail est l'utilisation des ontologies pour la description et la gestion des objets virtuels. Cela permet d'assurer la réutilisation, l'extensibilité et l'interopérabilité des données ce qui garantit l'orchestration efficace des objets pour former des collaborations. Cependant, les objets virtuels sont orchestrés par une entité centrale qui prend le contrôle sur l'ensemble du système. En plus, la solution qu'ils proposent est verticale et spécifique aux applications de gestion des incendies.

Le travail de [Kelaidonis et al., 2012] propose un cadre de gestion cognitive des objets IoT qui vise à fournir les moyens permettant de surmonter les problèmes d'hétérogénéité technologiques de l'IoT. L'architecture du cadre proposé comprend trois

niveaux d'abstraction :

- Objet virtuel (Virtual Object, VO) : qui donne des représentations virtuelles des différents objets du monde réel enrichies par des informations contextuelles.
- Objet virtuel composite (Composite Virtual Object, CVO) : qui définit une composition sémantiquement interopérable et cognitive de plusieurs VOs pour fournir des services de haut niveau qui répondent au mieux à des besoins complexes.
- Couche utilisateur : elle représente la partie hôte du système et l'interface via laquelle l'utilisateur aura le contrôle sur la gestion et la mise en œuvre des applications IoT.

Une grande partie de ce travail repose sur l'intervention de l'utilisateur vu que les représentations virtuelles sont utilisées d'une manière prédéfinie par l'utilisateur pour former des CVOs à valeur ajoutée afin de satisfaire leurs objectifs.

Le travail présenté dans [Mrissa et al., 2015] repose sur le concept d'*avatar* pour représenter des objets IoT physiques sur le Web. Dans ce travail, un avatar désigne un environnement d'exécution extensible et distribué qui s'appuie sur la sémantique et les protocoles du Web standards pour créer des applications WoT. Comme décrit, les avatars sont de nature active et leur architecture sous-jacente est basée sur une gestion centralisée. Dans cette dernière, les avatars reposent sur l'utilisation des requêtes-réponses sur un référentiel central qui contient toutes les descriptions sémantiques des avatars. Cependant très peu de détails sont fournis sur le fonctionnement général des avatars et la manière avec laquelle ils peuvent collaborer entre eux pour accomplir des applications complexes basée sur la composition de services. La solution proposée dans ce travail trouve ses limites dans des systèmes IoT distribués à grande échelle.

Le projet européen SENSEI, quant à lui [Presser et al., 2009][SENSEI, 2017], vise à créer une architecture ouverte qui concrétise la vision de l'Internet du monde réel (Real World Internet, RWI). Cette architecture est basée sur l'abstraction des réseaux de capteurs et actionneurs hétérogènes sans fil (WSAN), des dispositifs RFID ainsi que plusieurs autres catégories d'appareils intégrés au réseau. Les différentes catégories de dispositifs sont fournies par 19 partenaires de plusieurs sociétés de 11 pays européens qui se sont réunis au sein du consortium SENSEI. Ce projet utilise la notion de ressource pour désigner un objet virtuel. Une ressource SENSEI peut représenter un ou plusieurs objets. Elle expose leurs fonctionnalités via des interfaces universelles. Leur solution se limite à des représentations passives exploitées par des applications IoT généralement gérées au niveau du cloud et ne disposant d'aucune autonomie pour prendre des décisions.

IoT-A (IoT-Architecture) [De et al., 2011][Nitti et al., 2015] est un projet européen qui vise également à fournir une architecture de référence et un ensemble de blocs de base pour faciliter la mise en œuvre des applications IoT. Ce projet étend les modèles proposés dans le projet SENSEI en enrichissant les modèles de description des données produites ou utilisées par les objets physiques par des annotations sémantiques via

OWL-DL. Ces descriptions sémantiques incluent des informations contextuelles sur l'environnement de l'objet IoT ce qui donne un sens aux données brutes produites par les objets. Un VO, dans ce projet, est désigné par le terme entité virtuelle (Virtual Entity, VE) plutôt que par une ressource comme dans SENSEI. Un objet physique peut fournir un ou plusieurs services et chaque service est associé à une VE. Les blocs de construction de base de leur système sont des services de haut niveau prêts à l'emploi. Ils sont le résultat de l'orchestration de plusieurs VE. Le modèle proposé dans ce projet est très intéressant grâce à l'utilisation d'ontologies, mais les VE sont également des entités passives.

iCore [Kibria et al., 2016][Parodi et al., 2015][iCore, 2018] est un cadre qui exploite les capacités avancées de la modélisation sémantique du projet IoT-A pour représenter des objets du monde réel. La spécificité de l'architecture définie est qu'elle s'appuie sur un cadre cognitif pour automatiser le traitement des informations collectées et la prise des décisions correspondantes aux changements détectés. L'architecture iCore se compose de trois couches :

- Une couche basse appelée niveau d'objet virtuel (Virtual Object Level, VOL) fournit des représentations virtuelles des objets du monde réel qui peuvent être créés et détruits dynamiquement. Les capacités des objets sont universellement exposées aux couches supérieures et à d'autres architectures. Un VO peut faire abstraction de plusieurs objets réels et un objet réel peut être représenté par plusieurs VO comme dans l'IoT-A. Les VOs sont stockés dans un registre commun qui fournit des fonctions d'interrogation pour trouver le VO qui correspond le mieux à une demande.
- Une couche intermédiaire appelée niveau d'objet virtuel composite (Composite Virtual Object Level, CVOL) représente la couche qui fournit les mécanismes cognitifs pour le mashup sémantique des VOs disponibles pour créer des services complexes à valeur ajoutée. Ces mécanismes sont basés sur la logique événement/action. Les COVs construits sont également publiés dans un référentiel dédié.
- Une couche application appelée niveau de service (Service Level, SL) permet de traduire les demandes des utilisateurs finaux et rechercher les CVOs les plus appropriés pour les satisfaire.

La science cognitive et les mécanismes de raisonnement sémantique reflètent la force de l'architecture iCore, car ils automatisent le traitement des requêtes des utilisateurs. Cependant, les VOs n'ont aucune capacité de raisonnement autonome pour réagir aux événements internes ou externes, et ils sont simplement utilisés par la couche SL.

Une autre plateforme intéressante pour la virtualisation des objets IoT est ETSI M2M et son évolution vers le standard oneM2M. ETSI M2M [Lu et al., 2012a][Klinpratum et al., 2014] est un ensemble de spécifications techniques

développées en 2009 par l'institut européen des normes des télécommunications (European Telecommunications Standards Institute, ETSI) pour développer et maintenir des architectures IoT/M2M complètes en créant une couche de service commune. En 2012, l'ETSI s'est pleinement engagé dans l'initiative mondiale oneM2M constituée de sept organismes de normalisation internationaux [Alaya et al., 2015]. OneM2M vise à fournir une norme mondiale pour la construction des plateformes de services IoT. Dans cette norme, la virtualisation des objets physiques est représentée via des ressources RESTful. Une ressource est adressable de manière unique via un URI et elle correspond à un seul objet physique. Le travail de [Seydoux et al., 2016] vise à enrichir la description de ces ressources via une ontologie générique appelée IoT-O. Cette norme fournit des mécanismes permettant aux plateformes d'être hautement extensibles pour intégrer des technologies existantes et nouvelles. Elle fournit également des méthodes pour la découverte efficace des ressources via des mécanismes de requêtes et de filtres. Cependant, comme les autres travaux, ces ressources sont passives et ne peuvent pas être conscientes de leur état pour agir en conséquence.

2.1.1 Synthèse

Pour résumer, le Tableau 2.1 donne une synthèse des travaux étudiés selon les propriétés suivantes qui sont importantes et nécessaires pour répondre aux attentes des applications IoT :

- Cardinalité de la représentation : qui désigne le nombre d'objets assignés à une représentation virtuelle.
- Concept utilisé : qui correspond au nom donné à la représentation virtuelle sur le Web.
- Description sémantique : qui considère si le travail étudié s'appuie sur des descriptions sémantiques riches.
- Gestion des applications : qui indique si les applications IoT considérées sont gérées d'une manière centralisée ou décentralisée.
- Infrastructure de déploiement : qui définit l'environnement de déploiement de l'architecture.
- Type de comportement : qui correspond à un comportement passif ou actif de la représentation virtuelle.

Travail étudié	Cardinalité	Concept utilisé	sémantique	Gestion des applications	Déploiement	Comportement
[Römer et al., 2004]	Un à Plusieurs	Meta-contrepartie virtuelle (VMC)	Non	Centrale	Indéfini	Passif
[Han et al., 2013]	Un à Un	Profil de périphérique pour le service Web (DPWS)	Oui	Centrale	Serveur local	Passif
[Shamszaman et al., 2014]	Un à Un	Objet virtuel (VO)	Oui	Centrale	Indéfini	Passif
[Kelaidonis et al., 2012]	Un à Un	Objet virtuel (VO)	Non	Centrale	Indéfini	Passif
[Mrissa et al., 2015]	Un à Un	Avatar	Oui	Indéfini	Objet, Fog et Cloud	Actif
[SENSEI, 2017]	Un à Plusieurs	Ressource	Non	Distribuée	Cloud	Passif
IoT-A [De et al., 2011]	Plusieurs à Un	Entité virtuelle (VE)	Oui	Distribuée	Indéfini	Passif
[iCore, 2018]	Plusieurs à Plusieurs	Objet virtuel (VO)	Oui	Centrale	Fog et Cloud	Passif
ETSI M2M and oneM2M [Lu et al., 2012a]	Un à Un	Ressource	Oui	Distribuée	Fog et Cloud	Passif
Notre approche	Un à Un	Avatar	Oui	Distribuée	Objet, Fog et Cloud	Autonome et actif

TABLE 2.1 – Une synthèse des travaux étudiés

À travers cette étude comparative, nous avons constaté que la plupart des travaux existants autour de la virtualisation des objets IoT hétérogènes sont limités par rapport aux besoins variés des applications IoT et les caractéristiques omniprésentes des systèmes IoT complexes. Ces travaux se concentrent uniquement sur le problème d'interopérabilité dans leurs propositions de virtualisation. Ils proposent de représenter les objets IoT par des entités passives qui ont plusieurs dénominations (objet virtuel, ressource, avatar, etc.). Ces entités passives n'ont aucune capacité de raisonnement ou forme d'intelligence, mais elles sont uniquement utilisées par d'autres entités, généralement un gestionnaire central, pour la création des applications IoT et des services à valeur ajoutée. Par conséquent, ces propositions souffrent du problème de point de défaillance unique et en cas de panne du gestionnaire, l'ensemble du système devient défaillant. En plus, elles sont limitées dans le cadre des systèmes IoT à large échelle.

Pour cela, nous proposons dans ce travail d'étendre ces représentations virtuelles passives avec des capacités de raisonnement intelligent constituant ainsi des avatars autonomes. Suite à des événements internes du système ou externes, ces derniers sont capables de découvrir et de sélectionner les avatars qui peuvent collaborer avec eux pour accomplir des objectifs complexes (applications IoT), sans l'intervention d'une entité tierce centrale. Les avatars autonomes sont déployés dans une architecture distribuée de type fog/cloud computing. Cette architecture va constituer la base sur laquelle nous allons construire nos approches de découverte et de sélection des services IoT.

2.2 Contribution I : Architecture IoT distribuée basée sur des avatars autonomes

Dans cette section, nous détaillons notre première contribution qui consiste en une architecture IoT distribuée basée sur la notion d'avatar autonome.

2.2.1 Avatars : Définition et Caractéristiques

Un avatar ou un objet virtuel, les deux mots étant utilisés comme synonymes, est une entité qui se base sur le concept de virtualisation. Par conséquent, il est difficile de lui associer une définition précise non-ambiguë. Le flou autour de ce terme résulte de l'utilisation extensive du mot virtuel au cours des dernières décennies dans plusieurs contextes, tel que : les réseaux de communication.

Beaucoup de travaux comme [Vlacheas et al., 2013], [Kelaidonis et al., 2012], [Malik et al., 2019], [Christophe et al., 2011] et [Shamszaman and Ali, 2017] par exemple, ont emprunté le terme d'objet virtuel dans le domaine de l'IoT. Récemment, quelques contributions [Kuzminykh, 2018] [Médini et al., 2017] ainsi que le consortium de standardisation W3C [W3C, 2015] ont utilisé le mot *Avatar* comme une alternative du mot objet virtuel. Sans perdre de généricité, un avatar est défini comme une abs-

traction virtuelle et passive d'une entité physique ou logicielle sur le Web. Il permet d'exposer les différentes fonctionnalités fournies par l'entité représentée via des services Web standards. Cette abstraction a pour but de fournir une interface interopérable et standardisée d'une entité concrète. Dans ce travail, nous améliorons la définition initiale d'un avatar en le transformant d'un simple fournisseur de services en une entité autonome capable de raisonner sur ses propres connaissances, modélisées via des descriptions sémantiques, pour adopter un comportement situé selon le contexte dans lequel elle se trouve.

Un avatar de notre point de vue est défini comme une abstraction et une représentation numérique, sémantiquement enrichie, d'une entité du monde réel. Il est caractérisé par : sa capacité d'analyser et d'interpréter les données acquises de son environnement, sa capacité de communiquer directement avec les autres avatars, une vue et une perception partielle et limitée de son environnement, et un ensemble d'objectifs à satisfaire individuellement ou collectivement. Il possède les attributs suivants :

- URI (Uniform Resource Identifier) : il est utilisé pour attribuer une identité unique à un avatar et le rendre adressable sur le Web via un point d'accès.
- Ontologie : il contient une référence unique vers une instance de l'ontologie qui décrit l'avatar dans ses diverses dimensions : caractéristiques physiques, fonctionnelles et contextuelles.
- Services : ils représentent l'ensemble des capacités, décrites via des méthodes, des entrées et des sorties, de l'entité concrète représentée par l'avatar sur la Web. Ils sont modélisés via un vecteur $S_i = \{s_i^1, s_i^2, ..., s_i^q\}$, où : q est le nombre de services de l'avatar A_i.
- Fonctionnalités : elles représentent une description générique des services réalisable par l'objet IoT. Elles sont modélisées via un vecteur $f_i = \{f_i^1, f_i^2, ..., f_i^p\}$? où : p est le nombre de services de l'avatar A_i.
- Objectifs (applications IoT) : ils désignent l'ensemble d'applications complexes qu'un avatar peut accomplir en collaboration avec ses homologues. Un objectif est souvent activé d'une manière automatique une fois que ses préconditions soient satisfaites.
- Événements : ils fournissent les conditions préalables nécessaires au déclenchement d'un objectif donné.
- Contexte social : il est décrit avec un ensemble de métadonnées sémantiques qui permettent à un avatar de construire ses liens sociaux. Ces méta-données comprennent : **a)** un ensemble de centres d'intérêts quantifiés avec un degré d'intérêt pour chaque sujet. Les degrés d'intérêt d'un avatar A_i sont modélisés avec un vecteur $I_i = \{I_i^1, I_i^2, ..., I_i^m\}$ où m représente le nombre de sujets d'intérêts. Chaque composante de ce vecteur, c'est-à-dire le degré d'intérêt, est représentée avec une valeur réelle dans l'intervalle [0,1], **b)** une localisation définie avec la longitude

et la latitude (LongA_i, LatA_i) de l'objet IoT que l'avatar A_i représente, et **c)** un identifiant du propriétaire ou de groupe de propriétaires de l'objet représenté.

Les principaux avantages qui peuvent être tirés l'utilisation des avatars autonomes dans l'IoT sont :

- Interopérabilité : l'utilisation des avatars pour modéliser les différents objets IoT hétérogènes d'une manière uniforme et standard via les descriptions sémantiques, quelles que soient leur nature et leurs technologies de communication, permet de garantir l'interopérabilité et l'inter-fonctionnement des systèmes IoT.
- Autonomie : l'intégration de l'autonomie dans le comportement d'un avatar lui permet de raisonner et de prendre des décisions intelligentes sans l'intervention d'une entité tierce.
- Sensibilisation au contexte : la capacité d'un avatar à acquérir, à analyser et à interpréter des informations liées à son environnement lui permet s'adopter le comportement adéquat à la situation détectée.
- Découverte et sélection de services : pour accomplir ses objectifs complexes, l'avatar est capable d'achever le processus de découverte et de sélection des services qui répondent au mieux aux exigences fonctionnelles et aux attentes de QoS de ces objectifs.

2.2.2 Architecture générique d'un avatar

L'architecture générique d'un avatar est composée de trois composants comme illustré en Figure 2.1 que nous allons détailler dans les sous-sections qui suivent :

- Une base de connaissances (Knowledge Base, KB) qui contient les données de l'avatar et ses comportements modélisés via des règles logiques.
- Un composant objet virtuel (Virtual Object) passif qui garantit l'interopérabilité de l'objet IoT hétérogène représenté d'une manière uniforme et standardisée en utilisant les descriptions sémantiques.
- Un composant comportement autonome (Autonomous behavior) qui permet de modéliser le comportement autonome de l'avatar dans la prise de décisions.

Dans ce qui suit, nous allons développer les notions relatives aux descriptions sémantiques utilisées pour mettre en place la base de connaissance d'un avatar et son composant de virtualisation ainsi que les objectifs fonctionnels (applications IoT) qu'il fournit. Nous détaillons également les mécanismes de raisonnement qui permettent de doter l'avatar de son comportement autonome.

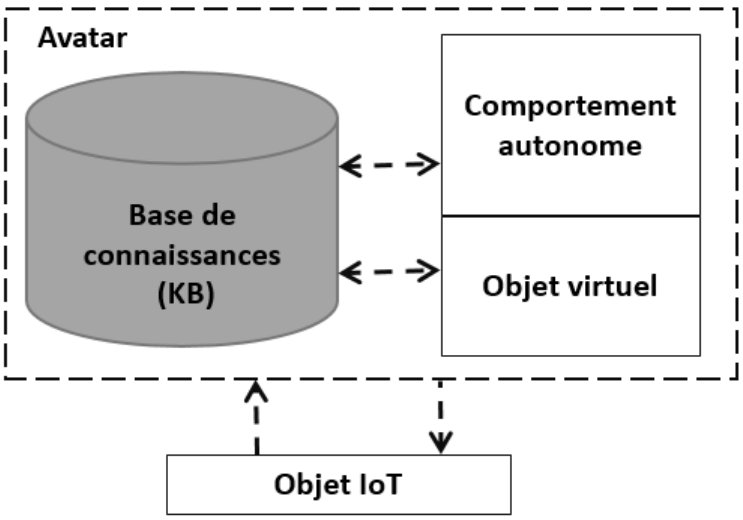


FIGURE 2.1 – Architecture générique d'un avatar

2.2.3 Base de connaissances (KB)

Une base de connaissances d'un avatar regroupe l'ensemble de ses connaissances relatives à son état interne et sa propre vision de son environnement indépendamment des connaissances des autres avatars de son système. Elle est structurée sous une forme exploitable. Pour cela, les descriptions sémantiques constituent un élément clé pour permettre à l'avatar d'interpréter ses données brutes pour les rendre compréhensibles. Elles sont utilisées également pour exposer leurs fonctionnalités et leurs services d'une manière standard sur le Web. De plus, des mécanismes d'inférence et de raisonnement peuvent être intégrés à la base pour tirer des conclusions et décider de l'action à effectuer en utilisant des règles de raisonnement. Dans le modèle proposé, la base de connaissances est constituée d'une ontologie générique instanciée appelée *AvatarOnt* et d'une série d'objectifs et de processus qui reflètent la partie fonctionnelle de l'avatar.

2.2.3.1 Ontologie *AvatarOnt*

L'ontologie *AvatarOnt*, illustrée en Figure 2.2, sert à fournir un vocabulaire non-ambigu et un modèle commun qui contient principalement une description de l'avatar et ses caractéristiques, les fonctionnalités qu'il peut réaliser et qui sont exposées en tant que services ainsi que les données qu'il produit ou les actions qu'il peut accomplir. Cette ontologie peut être réutilisée dans n'importe quel domaine IoT. Pour sa conception, nous nous sommes basés sur la méthodologie NeOn [NeOn, 2015] qui définit deux types d'exigences : conceptuelles pour présenter les concepts à intégrer dans l'ontologie à créer, et fonctionnelles quant à sa structure générale.

Les modules qui composent l'ontologie *AvatarOnt* sont :

- Un module de service : il est basé sur l'ontologie MSM pour décrire les opérations des services, leurs entrées et sorties, l'ontologie hRests pour les méthodes REST d'invocation de service ainsi que l'ontologie WSOnto pour exprimer leur partie non-fonctionnelle (QoS).
- Un module capteur : il est basé sur les ontologies SOSA et IoT-O pour décrire les capteurs et leurs observations.
- Un module actionneur : il est basé sur les ontologies SAN et IoT-O pour décrire les actionneurs et les actions à effectuer sur les appareils.
- Un module profil d'avatar : pour décrire le profil de l'avatar, ses objectifs, son nœud de déploiement et son contexte social.

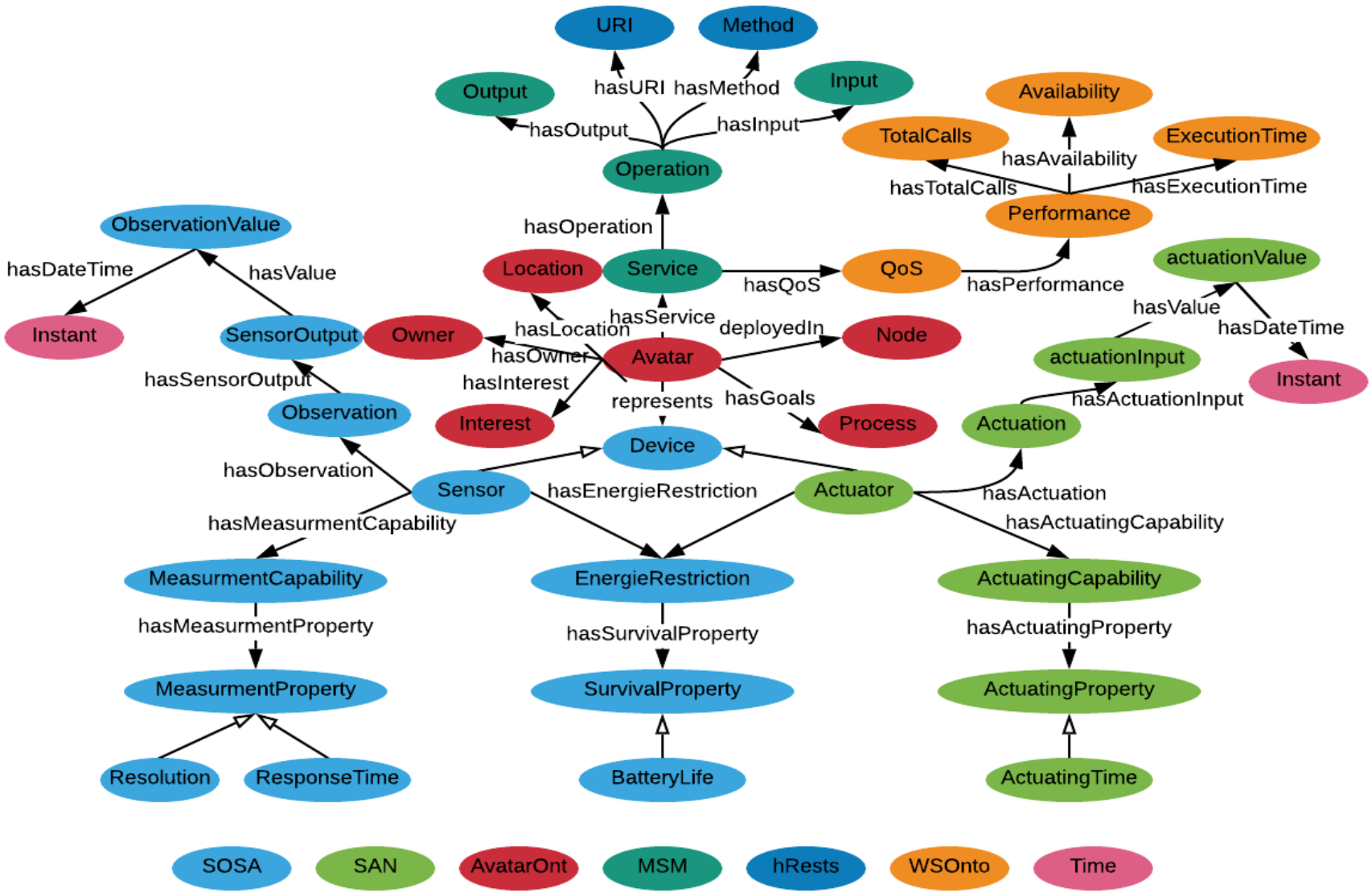


FIGURE 2.2 – Ontologie *AvatarOnt*

2.2.3.2 Objectifs fonctionnels (Applications IoT)

La base de connaissances contient également un ensemble d'objectifs à atteindre, modélisés sous forme de processus. Un processus est défini comme une série d'actions corrélées qui ont des dépendances entre elles au moment de l'exécution. Un objectif peut être décomposé en plusieurs tâches qui peuvent être complexes ou atomiques. Plusieurs approches ont été proposées à cet effet. La description abstraite du flux de travail par le concepteur est l'une des approches les plus courantes compte tenu de son coût réduit. Pour ce faire, nous utilisons une ontologie qui décrit les différentes tâches composites et atomiques qui composent l'application à réaliser et les dépendances logiques entre elles.

Nous étendons l'ontologie DEMISA proposée dans [Tietz et al., 2011] par le nœud *State* qui permet de désigner l'état du processus qui peut être : en attente, en cours d'exécution ou en pause, comme le montre la Figure 2.3.

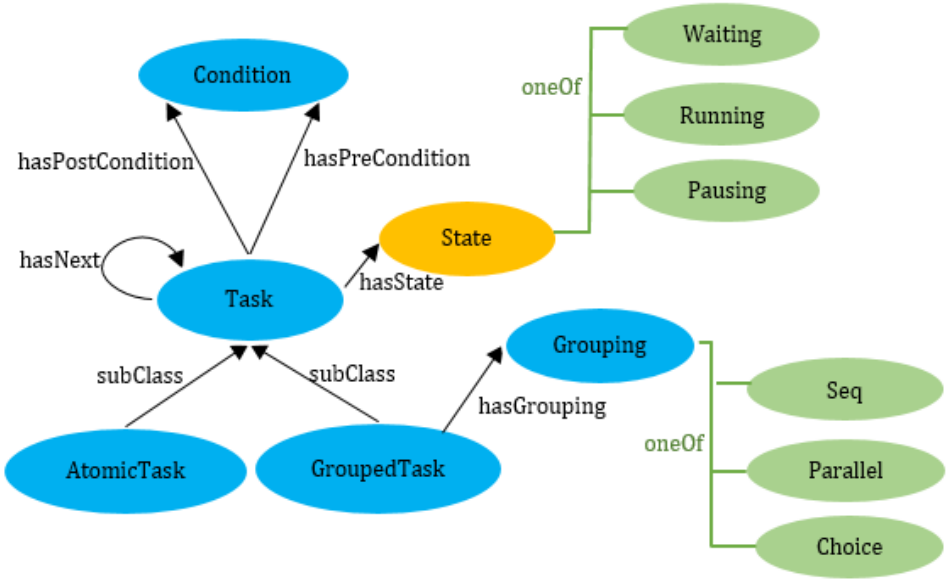


FIGURE 2.3 – Ontologie des objectifs DEMISA

L'ontologie indique qu'un processus est composé de plusieurs tâches qui peuvent être complexes ou atomiques. Les tâches complexes peuvent être de type séquentiel, parallèle ou de choix. Les tâches peuvent avoir des pré-conditions qui représentent les prédicats qui doivent être vrais pour que le processus soit exécuté ainsi que des post-conditions qui sont des propriétés qui doivent être vraies après l'exécution du processus.

Formellement, un processus est modélisé par un vecteur $T_{req} = \{T_1, T_2, ..., T_n\}$, tel que T_k est la $k^{ème}$ tâche du processus.

2.2.3.3 Mécanismes de raisonnement

Concernant les mécanismes de raisonnement, nous nous sommes tournés vers des méthodes basées sur des règles réactives pilotées par les événements, car ces méthodes ont prouvé leur efficacité dans des systèmes distribués déployés dans des environnements incertains, en particulier dans l'IA et les MAS. Le but de ce type de règles est de détecter des événements afin de permettre des réactions automatiques. Ils sont de type $E_1 \wedge E_2 \wedge ... \wedge E_n \rightarrow P_i$, où la première partie de la règle représente une conjonction (ou une disjonction) d'événements et la deuxième partie P_i représente le processus à accomplir à la suite d'événements.

Le raisonnement et la prise de décision au sein d'un avatar sont fournis essentiellement en se basant sur la base de connaissances et deux autres composantes comme illustré dans la Figure 2.4.

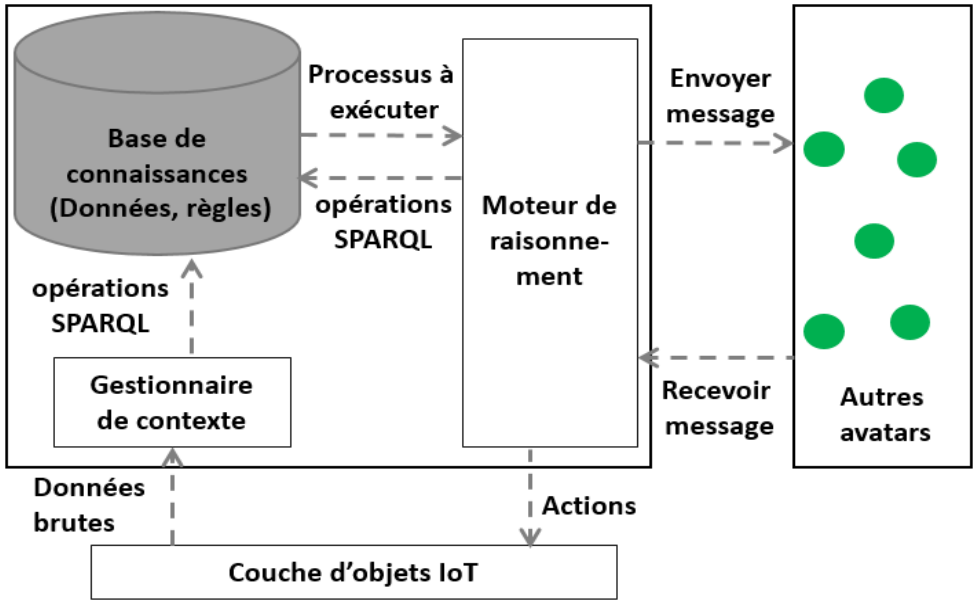


FIGURE 2.4 – Modules de raisonnement

- Gestionnaire de contexte (Context Manager) : il permet de surveiller en permanence les changements qui peuvent survenir dans l'environnement de l'avatar (obstacle devant un véhicule) ou au niveau de l'objet physique qu'il représente (son état ou ses données). Une fois qu'un changement est détecté, ce gestionnaire prend la responsabilité de mettre à jour la base de connaissances de l'avatar via les opérations SPARQL (création, mise à jour et suppression).
- Moteur de raisonnement (Reasoning Engine) : ce module intègre plusieurs algorithmes responsables de l'exécution des processus (pour atteindre un objectif)

déclenchés par des règles logiques, du traitement des messages reçus par l'avatar et de la mise à jour de sa base de connaissances. Dans le cas des processus complexes qui nécessitent la collaboration de plusieurs avatars, ce module s'occupe de la découverte et de la sélection basée sur la QoS des avatars qui peuvent être impliqués dans la réalisation du processus.

2.2.3.4 Application : modélisation du processus de dépassement et les avatars candidats

Le scénario de dépassement peut être représenté via un processus fonctionnel composé de plusieurs tâches élémentaires comme indiqué dans la Figure 2.5, tel que : X indique l'opérateur ET (AND), et modélisé en utilisant les descriptions sémantiques comme donné dans l'illustration A.1 en Annexe A.

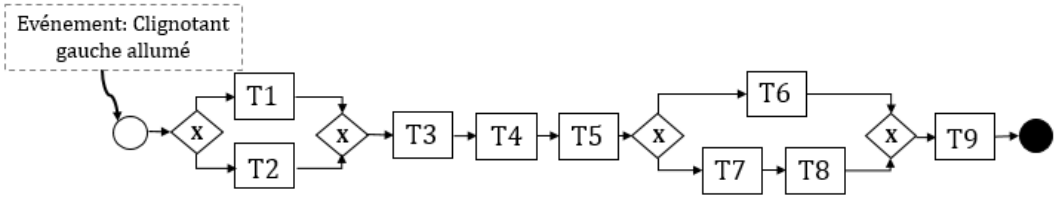


FIGURE 2.5 – Processus de dépassement de véhicules connectés

Le processus de dépassement est déclenché lorsque le conducteur du véhicule hôte allume le clignotement à gauche de son véhicule pour indiquer son intention de dépasser. Après cela, dans un premier lieu, le conducteur vérifie si le véhicule arrière a déjà entamé un dépassement (T1) et idem pour le véhicule à doubler (T2). Si c'est le cas, et si l'un des deux véhicules est entrain de dépasser, le dépassement n'est pas possible, sinon, d'autres conditions doivent être vérifiées. Tout d'abord, la distance du véhicule hôte par rapport au véhicule à doubler est mesurée (T3), et si elle est supérieure à 30 mètres, la vitesse du véhicule à doubler est également mesurée (T4). Ensuite, la différence de vitesse entre le véhicule hôte et le véhicule à doubler est calculée (T5). Si le véhicule hôte roule à 20 km/h au minimum plus vite que le véhicule à doubler, un dépassement de type accéléré est effectué. Afin de garantir cela, la distance disponible pour le dépassement est mesurée (T6) et la distance de sécurité minimale requise pour un dépassement en toute sécurité est calculée (T8) à l'aide des formules données dans [Mo et al., 2018]. Pour effectuer la tâche (T8), la vitesse du véhicule venant du sens inverse est requise (T7). Si la distance disponible est supérieure à la distance de sécurité minimale requise (T9), un message autorisant le conducteur à dépasser s'affiche.

Nous supposons que ce processus est effectué dans la situation illustrée dans la Figure 2.6.

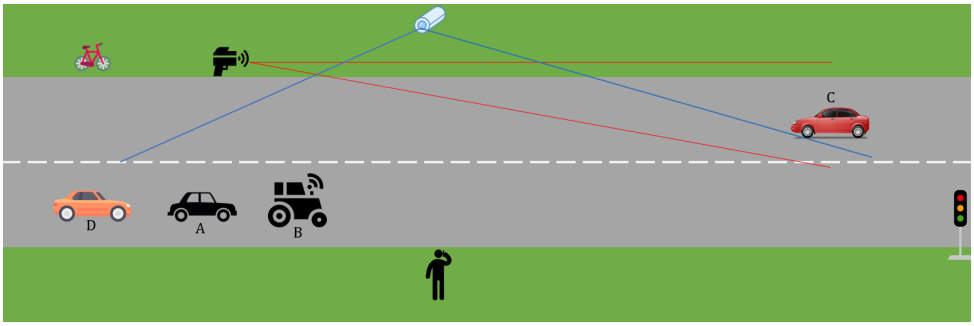


FIGURE 2.6 – Scénario de dépassement

Dans ce scénario, nous supposons que le véhicule hôte A veut dépasser le tracteur B qui roule à basse vitesse, et que ce véhicule n'a pas les mécanismes de perception nécessaires qui lui permettent de connaître son environnement. Par conséquent, pour effectuer le dépassement en toute sécurité, le véhicule A doit collaborer avec les appareils de son entourage. Nous retrouvons : un radar et une caméra installés au bord de la route, un vélo roulant à côté, un passager qui utilise un smartphone, des feux de signalisation à quelques mètres du véhicule hôte, les smartphones des conducteurs de A, B et C ont un système mondial de navigation par satellite (GNSS) utilisé pour le suivi des véhicules et un RSU (Road-Side Unit) qui couvre la région dans laquelle se trouve le véhicule hôte. Nous supposons également que la vitesse de déplacement des véhicules hôte, à doubler, derrière et le véhicule venant du sens inverse (A, B, D et C respectivement) est constante.

Un récapitulatif des avatars disponibles est donné dans le Tableau 2.2. Pour chaque avatar, ses services IoT, ses fonctionnalités, ainsi que leurs informations sociales (intérêts, location et propriétaire) sont donnés. Plusieurs services peuvent être fournis par le même avatar, par exemple pour l'avatar A_1 qui représente le véhicule hôte, il possède : un service pour mesurer la distance du véhicule par rapport à une cible qui se trouve derrière via un lidar intégré dans l'arrière du véhicule, un service mesurer la vitesse de cette cible, et un service de calcul pour effectuer des calculs mathématiques et logiques. Pour faciliter la compréhension de l'étape de découverte, qui permet de choisir les avatars qui offrent des services capables d'implémenter les tâches du processus, nous faisons directement l'association entre les services fournis par les avatars candidats et chaque tâche abstraite du processus de dépassement, tels que : S_{ij} signifie que l'avatar A_i peut effectuer la tâche abstraite T_j. Les intérêts sociaux sont : I_1 est le Transport, I_2 est la Météo, I_3 est la Perception d'environnement, I_4 est la Détection d'obstacles, I_5 est le Réseau informatique, I_6 est la Luminosité, I_7 est la Géolocalisation et I_8 est

la Circulation. Concernant les fonctionnalités, f_1 est le traitement d'images, f_2 est la Mesure de distance, f_3 est la Mesure de vitesse, f_4 est la Capacité de calcul, f_5 est le Déplacement et f_6 représente la Communication.

Avatar	Services	Intérêts	Localisation	Propriétaire	Fonctionnalités
A_1 : Véhicule A	S_{13},S_{14},S_{15}, S_{18},S_{19}	$I_1(0.6)$, $I_2(0.2)$, $I_4(0.6)$	43.57834 1.441185	$User_1$	f_2, f_3, f_4, f_5, f_6
A_2 : Véhicule B	S_{22}, S_{24}, S_{25}, S_{28}, S_{29}	$I_1(0.9)$, $I_2(0.3)$, $I_3(0.5)$, $I_4(0.6)$	43.578291 1.441146	$User_2$	f_2, f_3, f_4, f_5, f_6
A_3 : Véhicule C	S_{35}, S_{38}, S_{39}	$I_1(0.9)$, $I_2(0.3)$, $I_3(0.4)$, $I_4(0.5)$	43.578074 1.441052	$User_3$	f_2, f_3, f_4, f_5
A_4 : Véhicule D	S_{41}, S_{45}, S_{47}, S_{48}, S_{49}	$I_1(0.9)$, $I_2(0.3)$, $I_3(0.4)$, $I_4(0.5)$	43.578074 1.441052	$User_4$	f_2, f_3, f_4, f_5, f_6
A_5 : Vélo	/	$I_1(0.9)$	43.578315 1.441277	$User_5$	f_5, f_6
A_6 : RSU	/	$I_1(0.7)$, $I_5(0.7)$, $I_7(0.5)$	43.577163 1.440768	$User_6$	f_5, f_6
A_7 : Radar	S_{73}, S_{74}, S_{76}, S_{77}	$I_1(0.7)$, $I_4(0.9)$	43.57827 1.4412062	$User_6$	f_2, f_3, f_5
A_8 : Caméra	S_{81}, S_{82}, S_{83}, S_{86}	$I_1(0.6)$, $I_2(0.4)$, $I_3(0.9)$, I4(0.7)	43.578405 1.441300	$User_6$	f_1, f_2, f_5
A_9 : Feu de circulation	/	$I_1(0.7)$, $I_8(0.9)$	43.577899 1.440873	$User_6$	f_5
A_{10} : GNSS	S_{101}, S_{102}, S_{103}, S_{104}, S_{106}, S_{107}	$I_1(0.8)$, $I_4(0.7)$, $I_7(0.9)$	Serveur	$User_7$	f_1, f_2, f_3, f_5
A_{11} : Éclairage routier	/	$I_1(0.2)$, $I_4(0.3)$, $I_6(0.9)$, $I_8(0.4)$	43.57822 1.441251	$User_6$	f_5
A_{12} : Smartphone A	S_{123}, S_{125}, S_{128}, S_{129}	$I_1(0.4)$, $I_5(0.9)$, $I_7(0.7)$	43.57834 1.441185	$User_1$	f_4, f_6
A_{13} : Smartphone B	S_{132}, S_{134}, S_{135}, S_{138}, S_{139}	$I_1(0.4)$, $I_5(0.9)$, $I_7(0.7)$	43.578291 1.441146	$User_2$	f_4, f_6
A_{14} : Smartphone C	S_{147}	$I_1(0.4)$, $I_5(0.9)$, $I_7(0.7)$	43.578074 1.441052	$User_3$	f_4, f_6
A_{15} : Smartphone X	S_{151}, S_{152}, S_{156}, S_{157}	$I_1(0.4)$, $I_5(0.9)$, $I_7(0.7)$	43.578332 1.441082	$User_8$	f_4, f_6
A_{16} : Caméra	S_{161}, S_{162}, S_{163}, S_{166}, S_{167}	$I_1(0.6)$, $I_2(0.4)$, $I_3(0.9)$, I4(0.7)	43.49881 1.441300	$User_6$	f_1, f_2, f_5
A_{17} : Lidar	S_{174}, S_{176}	$I_1(0.7)$, $I_4(0.9)$	43.57827 1.4412062	$User_6$	f_2, f_3, f_5
A_{18} : Smartphone D	S_{181}	$I_1(0.4)$, $I_5(0.9)$, $I_7(0.7)$	43.578074 1.441052	$User_8$	f_4, f_6

TABLE 2.2 – Modélisation des objets IoT candidats pour le dépassement

Nous donnons dans l'illustration A.2 en Annexe A la description sémantique de l'avatar A_1 du véhicule hôte comme exemple de modélisation d'avatars.

2.2.4 Architecture de déploiement des avatars

Dans les architectures traditionnelles de déploiement, les contreparties virtuelles sont utilisées plutôt par une entité tierce qui gère l'ensemble du système et prend en charge la découverte et la sélection des services pour la mise en place des applications IoT d'une manière centralisée. Par conséquent, si une défaillance se produit au niveau de cette entité ou au niveau du serveur de déploiement, qui se base généralement sur un environnement cloud, l'ensemble du système devient défaillant. De plus, il existe plusieurs limitations en raison de la distance entre les objets IoT et les centres de traitement cloud, notamment en termes de latence, de bande passante et de sécurité.

Nous proposons donc de déployer les avatars autonomes dans une nouvelle architecture fog/cloud computing distribuée. En effet, les avatars peuvent être déployés directement au niveau des objets s'ils disposent des ressources nécessaires, ou au niveau des nœuds fog relativement proches des utilisateurs finaux ou au niveau du cloud. L'architecture proposée est composée de deux couches logiques comme le montre la Figure 2.7.

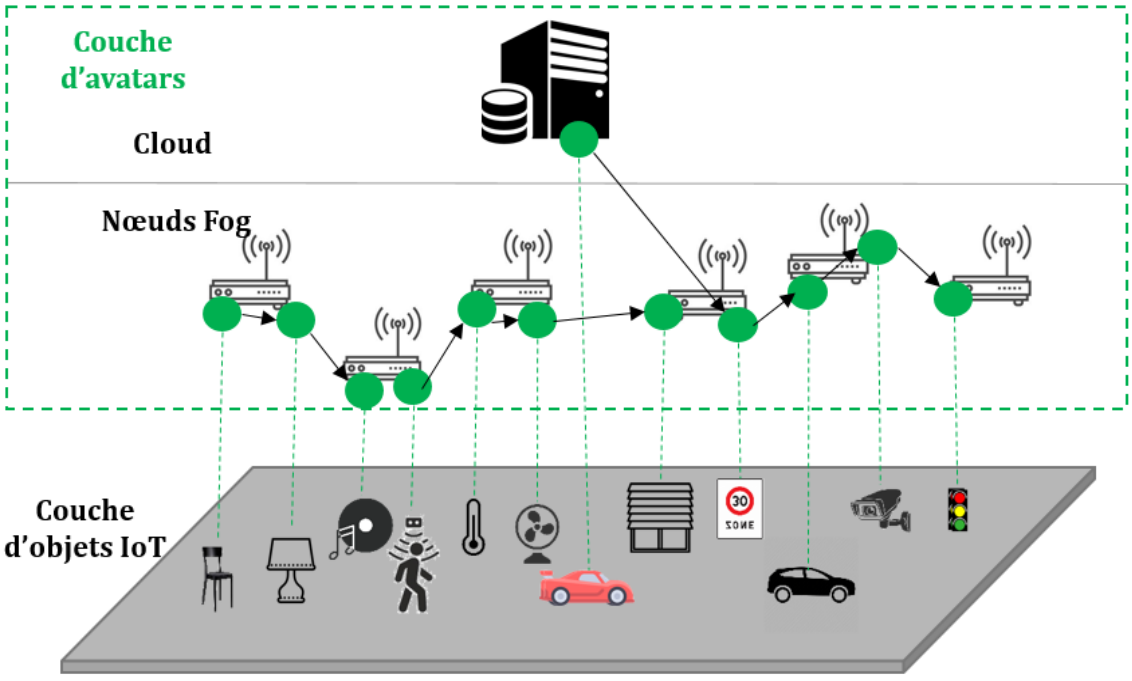


FIGURE 2.7 – Architecture IoT basée sur les avatars

- Couche d'objets physiques (Devices Layer) : elle représente une collection d'objets du monde réel : capteurs, actionneurs, etc. Ces objets sont connectés via plusieurs

technologies à plusieurs passerelles qui sont à leur tour enregistrées auprès d'un serveur central qui se charge de stocker les données à long terme pour des fins d'analyses stratégiques et qui donne l'accès à l'ensemble du système. Ces appareils sont hétérogènes, ils peuvent être mobiles ou fixes, comme ils peuvent être déployés dans des environnements distribués.

- Couche d'avatars (Avatars Layer) : chaque objet de la couche d'objets est représenté via un avatar qui virtualise son profil et ses fonctionnalités à l'aide des descriptions sémantiques. Ces avatars sont des entités actives. Ils ont un comportement autonome qui leur permet de réagir suite à des événements modélisés par des changements de leur contexte ou leurs données internes. Les capacités de réaction autonomes sont liées à l'utilisation des règles réactives et des mécanismes de raisonnement avancés. Les avatars peuvent également communiquer via des messages (sous forme de requêtes) et collaborer entre eux afin de résoudre des problèmes communs d'une manière distribuée et autonome.

Dans ce travail, nous faisons abstraction de la partie réseau relative au déploiement des avatars dans les nœuds fog et dans le cloud, et la gestion de leur migration entre les nœuds avec la mobilité. Ces aspects sont traités par d'autres thèses dans notre équipe [Djemai et al., 2019] [Djemai et al., 2020] [Stypsanelli et al., 2020].

2.3 Conclusion

Dans ce chapitre, nous avons présenté notre première contribution qui consiste en une architecture modulaire qui repose sur un artefact logiciel appelé *avatar autonome* pour la fourniture des services à valeur ajoutée. Un avatar n'est qu'une contrepartie virtuelle d'une entité physique ou logicielle sur le Web doté d'un comportement autonome. Ce modèle architectural est adapté pour le développement de systèmes IoT complexes et interopérables. Il offre l'opportunité de dépasser les limites des œuvres existantes qui reposent sur un contrôleur central qui manipule et gère les avatars passifs.

Nous avons tout d'abord présenté une synthèse des principaux travaux autour de la virtualisation des objets IoT hétérogènes. Ensuite, nous avons donné la définition et les caractéristiques d'un avatar ainsi que son architecture et ce qu'il apporte pour l'IoT. Enfin, l'architecture IoT générique basée sur les avatars est donnée.

Cette première contribution fournit une base et un support pour mettre en place des mécanismes de découverte et de sélection des services fournis par les différents avatars qui peuvent collaborer ensemble pour accomplir des objectifs complexes. Le mécanisme de découverte proposé dans le chapitre prochain est basé les réseaux sociaux et sur les méthodes de clustering pour garantir une découverte évolutive via la réduction l'espace de recherche.

Chapitre 3

Vers une découverte distribuée basée sur les réseaux sociaux et le clustering

Sommaire

Avec l'explosion du nombre d'objets IoT, l'identification et la découverte d'un service IoT particulier deviennent un problème difficile. Pour cela, nous proposons dans ce chapitre une nouvelle approche de découverte basée sur des mécanismes de réseautage social (Social Networking, SN) et des algorithmes de clustering. L'utilisation de relations sociales permet d'améliorer l'efficacité et l'évolutivité du processus de découverte. Elle permet également d'accélérer et de faciliter la navigabilité dans un réseau IoT dynamique composé d'un grand nombre d'objets. Pour rendre la découverte encore plus efficace et pour réduire l'espace de recherche, une méthode de clustering basée sur l'algorithme fuzzy c-means est utilisée pour classer les objets dans le réseau social selon leurs fonctionnalités. Cette méthode permet aussi de bien guider la transmission des requêtes aux avatars les plus appropriés.

Tout au long de ce chapitre, nous allons tout d'abord donner un aperçu sur les concepts relatifs au paradigme des réseaux sociaux. Ensuite, un panel d'approches sur

la découverte des services basée sur les réseaux sociaux est présenté. Après cela, notre contribution concernant la découverte distribuée des services IoT basée sur le réseautage social et les mécanismes de clustering est détaillée.

3.1 Aperçu sur les réseau sociaux

Les réseaux dits *sociaux* ont été initialement introduits par l'anthropologue australien John Arundel Barnes en 1954 [Barnes, 1954] pour qu'ils soient utilisés dans le domaine des sciences humaines et sociales. Dans ce domaine, les réseaux sociaux désignent simplement un ensemble d'individus (ou d'entités au sens large) reliés entre eux selon des engagements et des critères de similarité tels que les origines et les centres d'intérêts. Globalement, un réseau social peut se définir comme « un ensemble d'individus, d'organisations, ou d'entités entretenant des relations sociales fondées sur l'amitié, le travail collaboratif et l'échange d'information » [Garton et al., 1997].

L'usage habituel de l'expression *réseaux sociaux* se réfère généralement aux réseaux sociaux numériques, qui recouvrent l'ensemble des plateformes de services permettant aux individus d'entretenir des interactions et des conversations sociales sur Internet. Andreas Kaplan et Michael Haenlein ont défini les réseaux sociaux numériques ou les médias sociaux comme « un groupe d'applications en ligne qui se fondent sur la philosophie et la technologie du net et permettent la création et l'échange du contenu généré par les utilisateurs » [Deighton et al., 2011].

Ces dernières années, beaucoup d'attention est donnée au sujet de l'intégration des concepts de sociabilités dans le domaine de l'IoT et le WoT permettant ainsi de transformer les objets IoT en objets sociaux. Cette convergence est souvent appelée soit l'Internet social des objets (Social Internet of Things, SIoT) [Atzori et al., 2012] ou le Web social des objets (Social Web of Things, SWoT) [Formo, 2012], deux termes émergents. Le SIoT ou le SWoT font référence à un réseau d'objets intelligents capables de s'organiser entre eux pour former des communautés en créant des relations sociales comme le font les humains. Les objets connectés auront l'habilité de créer des liens sociaux d'une manière autonome pour faire preuve d'un comportement collectif dans la résolution des problèmes complexes. Ce paradigme permet à un objet donné de découvrir d'une manière autonome les objets fournissant les services et disposant des caractéristiques souhaitées en utilisant ses relations avec ses voisins et les voisins de ses voisins d'une manière distribuée pour une découverte efficace et évolutive. Il facilite également la navigabilité dans un réseau dynamique constitué d'un très grand nombre d'objets voire des milliards d'objets.

3.1.1 Caractéristiques

Les réseaux sociaux peuvent être caractérisés par trois principales propriétés :

- Un ensemble d'acteurs, qui peuvent être des individus, des entités, groupes ou mêmes des organisations, rattachés par des interactions sociales qui définissent le type du lien social qui les relie.
- Des relations sociales qui peuvent être de différents types : familiales, professionnelles, régionales, etc. Les types de relations reliant les objets IoT sont détaillés dans la section suivante.
- Degré de séparation : cette caractéristique est connue sous l'expression *effet du petit monde* dans le domaine de la sociologie. Elle est basée sur l'hypothèse que la longueur des liens sociaux entre deux entités est généralement courte. Selon une étude réalisée par le psychologue Stanley Milgram en 1967 [Travers and Milgram, 1967], le degré de séparation maximale qui peut séparer la chaîne de connaissance de deux entités sociales choisies arbitrairement est limité à six.

3.1.2 Types des relations sociales dans l'IoT

Les relations d'interdépendance et les modèles d'interaction entre les objets IoT peuvent être directement inspirés des modèles sociologiques des individus humains. Les regroupements sociaux des objets communicants peuvent être guidés par plusieurs critères tels que : leurs intérêts en commun, des avantages mutuels qu'ils peuvent échanger ainsi que leur localisation géographique. Le modèle proposé dans [Atzori et al., 2014] constitue l'un des principaux modèles de la littérature. Dans ce modèle, les auteurs ont défini cinq types de relations sociales :

- Relation parentale des objets (Parental Object Relationship, POR) : ce type de relation est établi entre des objets homogènes issus d'un même fabricant et appartenant au même lot de production.
- Relation de co-localisation des objets (Co-Location Objects Relationship, C-LOR) : cette relation est établie entre des objets qu'ils soient homogènes ou hétérogènes situés dans la même zone géographique.
- Relation de travail des objets (Co-work Object Relationship, C-WOR) : la relation de travail est établie entre deux ou plusieurs objets coopérants pour réaliser un objectif commun.
- Relation des propriétaires des objets (Ownership Object Relationship, OOR) : il s'agit d'une relation englobant les objets appartenant au même propriétaire ou groupe de propriétaires.
- Relation sociale des objets (Social Object Relationship, SOR) : c'est une relation qui se produit lorsque les objets entrent en contact de façon sporadique ou continue lorsque leurs propriétaires se croisent, tels que : les camarades de la même classe ou bien les membres de la même famille.

En plus des relations sociales citées ci-dessus, le travail [Ali et al., 2018] présente quelques relations supplémentaires adaptées et appropriées à certains scénarios distincts :

- Relation fraternelle d'objets (Sibling Object Relationship, SIBOR) : cette relation englobe les objets appartenant à la même famille ou au même groupe d'amis. Ce type de relation est établi lorsque la confiance devient un facteur et une composante importance pour l'élaboration d'un lien social.
- Relation d'objets invités (Guest object relationship, GSTOR) : elle est créée entre les objets des utilisateurs qui jouent le rôle d'invité lorsqu'ils se rendent chez leurs amis ou leurs voisins. Ce type de relation leur donne le privilège d'accéder à plusieurs types d'informations qui sont restreintes à leurs proches.
- Relation d'objets étrangers (Stranger Object Relation, STGOR) : cette relation est formée entre les objets qui se croisent régulièrement dans des endroits spécifiques, tels que : les routes et les environnements publics. Ces objets sont anonymes et ne se connaissent pas forcement à priori.
- Relation des services d'objets (Service Object Relationship, SVOR) : ce type de relation s'applique aux objets dont les services font partie de la même composition de services pour répondre à un besoin utilisateur complexe.

3.1.3 Avantages des réseaux sociaux dans l'IoT

L'application des principes de réseautage social à l'IoT peut entraîner d'importants bénéfices :

- Navigabilité du réseau SWoT : à partir d'un seul objet, les autres appareils du réseau peuvent être atteints et parcourus en exploitant les différents liens sociaux qui les interconnectent et les informations locales qui sont stockées au niveau des objets.
- Scalabilité : elle désigne la capacité du réseau IoT à s'adapter à l'évolutivité accrue du nombre d'objets connectés et à maintenir son fonctionnement tout en gardant ses performances en cas de forte demande. Cette caractéristique est garantie dans le paradigme du SWoT grâce au principe de navigabilité qui permet de considérer un nombre important d'objets sans se soucier des performances vu que les traitements sont effectués d'une manière distribuée et collaborative entre les objets.
- Découverte distribuée des services : le SWoT possède le potentiel de fournir une infrastructure appropriée pour l'approvisionnement et la découverte efficace des services fournis par les divers objets IoT. Cela permet de mener des applications complexes d'une manière collaborative et évolutive. La découverte efficiente des services est essentiellement approuvée grâce aux mécanismes de navigabilité dis-

tribuée sur un réseau social, contrairement aux approches traditionnelles typiques de type centralisée qui ne sont plus adaptées aux milliards d'objets futurs.

3.2 Découverte des services basée sur les réseaux sociaux : un panel de contributions

Si les approches traditionnelles de découverte des services incluent un nombre très important de recherches scientifiques et de travaux de développement, les approches basées sur le réseautage social sont relativement nouvelles.

Le travail de [Kranz et al., 2010] propose d'utiliser les réseaux sociaux humains comme une infrastructure afin de permettre aux utilisateurs de partager les services fournis par leurs objets avec leurs amis. Cependant, cette solution implique une intervention humaine forte et s'oppose au principe de connectivité sur lequel se base l'IoT.

Le travail de [Maamar et al., 2011] s'intéresse également à l'utilisation de la métaphore des réseaux sociaux dans le domaine des services Web. Les auteurs se sont inspirés des réseaux sociaux humains conventionnels comme Facebook pour proposer un nouveau cadre permettant la découverte distribuée des services Web nommé LinkedWS (Linked Web Services). Ce modèle se base sur le principe de la capture des différentes interactions qui peuvent se produire entre les services Web pour créer en conséquence des relations sociales entre eux. Ces relations peuvent être principalement de type collaboration ou substitution. Cette méthode est intéressante, cependant, les réseaux sociaux sont construits en fonction des interactions entre des services, i.e., le degré d'interaction, ce qui ne reflète pas réellement l'aspect social.

Dans [Fallatah et al., 2014], une nouvelle approche pour la publication et la découverte des services Web est présentée. Elle est basée sur l'idée de la prise en compte des différents utilisateurs et les services Web sur le même réseau social mondial pour examiner l'influence des uns sur les autres en considérant des relations de type utilisateur-service, utilisateur-utilisateur et service-service. Cette méthode permet de bénéficier de ces liens créés pour améliorer la publication des services pour les rendre visibles auprès des utilisateurs intéressés, et par conséquent, de faciliter la recherche et la découverte des services pertinents. Néanmoins, l'idée de combiner les utilisateurs et les services Web sur le même réseau social mondial nécessite d'énorme ressources informatiques pour le concrétiser réellement vu le nombre important d'utilisateurs et de services Web. En plus, cette approche est beaucoup plus intéressante dans les domaines de l'e-commerce par rapport au domaine de l'IoT qui est basé sur l'interconnexion entre objets IoT et où les utilisateurs jouent un rôle relativement minime.

Dans le travail de [Nitti et al., 2014], les auteurs proposent une nouvelle approche permettant l'intégration des mécanismes de réseaux sociaux dans la mise en œuvre des solutions IoT. Dans cette approche, chaque nœud qui représente un objet IoT a la

possibilité de construire ses relations sociales d'une manière autonome en fonction de plusieurs règles bien spécifiées et définies par le propriétaire. Elle est basée sur l'idée d'utiliser la caractéristique de l'effet de petit monde pour permettre à un objet IoT de découvrir un service donné en explorant ses liens sociaux d'une manière distribuée. Pour la propagation et la transmission de sa requête, l'objet doit choisir parmi ses amitiés, le nœud avec la plus grande utilité. Pour ce faire, l'objet se base sur le degré de centralité de leurs voisins, i.e., le nombre de leurs connexions ce qui permet de mieux transmettre la requête. Cependant, l'utilisation de degré de centralité comme seul critère social pour la découverte des services n'est pas suffisant et n'est pas signifiant parce qu'il ne reflète pas l'importance et le profit en terme de fonctionnalités et de centres d'intérêts, mais plutôt uniquement en terme de diffusion.

Atzori and al. [Atzori et al., 2011] ont contribué à l'utilisation des réseaux sociaux dans l'IoT avec un modèle intéressant pour la découverte des services IoT. Ils ont proposé cinq types de relations, C-WOR, SOR, C-LOR, OOR et POR, qui ont été largement réutilisés dans plusieurs travaux ultérieurs. Le modèle architectural qu'ils proposent comprend essentiellement trois gestionnaires :

- Gestionnaire de relations sociales : il se base sur des règles bien décrites par les concepteurs pour créer, supprimer et mettre à jour des liens sociaux entre les objets IoT.
- Gestionnaire de découverte de services : il permet d'utiliser les relations sociales liant les objets IoT entre eux pour localiser les services requis pour accomplir un besoin applicatif donné.
- Gestionnaire de composition de services : une fois que les services requis pour répondre aux besoins d'une application sont découverts, ce gestionnaire se charge de les composer les uns avec les autres d'une manière réactive ou proactive selon des règles fixées. Il permet de gérer les interactions entre eux au moment de l'exécution.

Ce travail est intéressant, néanmoins, il est limité à une modélisation architecturale générique du système IoT combiné avec les concepts des réseaux sociaux. Les auteurs n'ont donné aucun détail concernant la découverte des services IoT et ils se sont contentés de brèves descriptions. En outre, leur système est basé sur les trois gestionnaires centraux ce qui le rend défaillant et non-évolutif.

Chen et al. dans [Chen et al., 2013], ont présenté une approche pour la découverte de services IoT qui tient compte des paramètres de QoS utilisés comme un type de relation, mais ne tient pas compte des relations de collaboration et de colocation entre les services qui s'avèrent importantes dans le domaine de l'IoT.Les propriétés QoS représentent un critère important, cependant, il n'est pas suffisant pour représenter les aspects sociaux d'un service IoT.

Dans [Hussein et al., 2015] [Hussein et al., 2017], un nouveau système appelé DSSoT (Dynamic Social Structure of Things) est proposé. Il s'agit d'une approche

basée sur le raisonnement cognitif pour la découverte des services d'une manière dynamique dans les environnements IoT. Leur approche s'appuie sur le principe de raisonnement sur les besoins "situationnels" à court terme des utilisateurs, leurs préférences ainsi que les environnements dans lesquels ils se trouvent pour générer la liste des objets disponibles fournissant les services qui peuvent accomplir les besoins requis et qui sont adaptés aux situations présentes. Pour la découverte de ces services, un algorithme d'appariement sémantique est utilisé pour chercher des correspondances entre les besoins utilisateurs et les services IoT disponibles en se basant sur un gestionnaire central. Ce travail illustre la mise en œuvre de la découverte dynamique des services intelligents dans un scénario de la vie réelle, en prenant comme exemple une instance de terminal d'aéroport, mais il n'a pas été étendu à d'autres instances dans d'autres domaines.

Un algorithme de découverte des services dans le contexte des systèmes de gestion d'objectifs (Goal management system) est présenté dans [Kasnesis et al., 2017]. Il englobe trois composants intelligents : a) un composant pour la découverte des services, b) un composant pour l'orchestration des services, et c) un dernier composant pour la liaison des services (binding). Selon l'algorithme proposé, l'objet commence tout d'abord par chercher les services requis dans la liste de ses amis, s'il ne trouve aucun service disponible, il lance le processus de découverte en choisissant l'une de ces trois méthodes :

- Approche basée sur les relations sociales : elle est basée sur l'exploration de son réseau social en profondeur, i.e. chercher le service dans la liste des amis puis les amis des premiers amis et ainsi de suite.
- Approche basée sur la région géospatiale : elle permet de chercher le service requis par régions géospatiales.
- Approche basée sur les réseaux MANET : elle permet d'exploiter la nature distribuée et dynamique des MANETs par l'utilisation de plusieurs techniques (Tradisional flooding, Single Random walkers, etc.) pour la transmission de la requête de découverte.

Le grand inconvénient de cette approche est qu'elle est basée sur un composant central pour la découverte des services et sur le mécanisme d'orchestration ce qui rend le système non-flexible et sujet aux pannes : dans le cas où le composant de découverte et d'orchestration tombe en panne, l'ensemble du système devient non fonctionnel.

Plusieurs plateformes SoWoT ont été proposées à cet issu, dont les plus importantes sont : ThingSpeak, Lysis et Paraimpu.

ThingSpeak [thingspeak, 2019, Girau et al., 2013] représente l'une des premières plateformes SoWoT et qui a été utilisée dans plusieurs domaines, tels que : l'éducation et l'agriculture. Elle fournit des fonctionnalités pour la création et la gestion des relations sociales entre les objets IoT incorporées à un serveur centralisé en se basant sur la

virtualisation de ces objets. Cependant, l'utilisation de la virtualisation est limitée à des enregistrements au serveur distant et ne fournit aucune autonomie en terme de création et de gestion de relations sociales. Ces dernières sont gérées au niveau du serveur central. Par conséquent, elle subit les inconvénients des approches centralisées.

Pour Lysis [Girau et al., 2016], il s'agit d'une plateforme SoWoT basée sur l'infrastructure cloud. Elle est déployée comme étant une plateforme de services (Platform-as-a-Service, PaaS). Dans ce travail, les auteurs se sont concentrés sur la manière de déployer les applications et les représentations des objets IoT dans le cloud. Néanmoins, les mécanismes de découverte des services fournis par les objets IoT, basés sur les réseaux sociaux, ne sont pas du tout abordés.

En ce qui concerne Paraimpu [Pintus et al., 2012], c'est une plateforme sociale qui permet d'inter-connecter les objets IoT physiques et les entités virtuelles sur le Web. Cependant, l'aspect social concerne le partage des objets IoT par les utilisateurs humains entre eux en s'appuyant sur les réseaux sociaux traditionnels comme Facebook et Twitter. Par conséquent, elle n'est pas adaptée au domaine de l'IoT.

3.2.1 Synthèse

Après avoir présenté les différents travaux de découverte basés sur les réseaux sociaux, nous présentons dans cette partie un récapitulatif des approches étudiées selon plusieurs critères comme indiqué dans le Tableau 3.1. Les critères considérés dans notre étude sont :

- **Architecture :** ce critère permet de définir l'architecture de déploiement du système utilisée dans l'approche étudiée.
- **Métriques sociales :** il désigne les métriques sociales et les types de relations exploitées pour créer le réseau social d'un objet IoT pour permettre la découverte.
- **Topologie :** il permet d'identifier la manière avec laquelle le système est géré, qui peut être centralisée ou distribuée.
- **Évolutivité :** ce critère désigne la capacité du système à s'adapter et continuer à fonctionner normalement en cas de passage à large échelle avec un nombre très important d'entités. Il est utilisé pour indiquer si l'approche étudiée est applicable aux systèmes à grande échelle.
- **Sémantique :** il fait référence à l'utilisation des spécifications sémantiques pour la description des différents objets IoT hétérogènes ainsi que les services qu'ils fournissent.
- **Processus de découverte :** il spécifie si le processus de construction du réseau social et les méthodes de découverte des services sont étudiées et prises en charge dans le travail étudié.

Référence	Architecture	Métriques sociales	Topologie	Évolutivité	Sémantique	Processus de découverte
[Kranz et al., 2010]	Architecture publication/Souscription	Relations entre les propriétaires d'objets	Distribuée	Non-défini	Non	Non
[Maamar et al., 2011]	Architecture UDDI	Degré d'interaction entre WS	Centralisée	Non-défini	Non	Oui
[Fallatah et al., 2014]	Architecture publication/Souscription	Relations entre propriétaires et WS	Centralisée	Non	Non	Non
[Nitti et al., 2014]	Architecture serveur/objets	Degré de centralité des objets	Distribuée	Oui	Non	Non
[Atzori et al., 2011]	Architecture serveur/objets	C-WOR, SOR, C-LOR, OOR, POR	Centralisée (multi-gestionnaires)	Non	Non	Non
[Chen et al., 2013]	Architecture UDDI	Propriétés non-fonctionnelles QoS	Centralisée	Non	Non	Non
[Hussein et al., 2015] [Hussein et al., 2017]	Architecture serveur/objets	personnalité, proactivité, localité, confiance	Centralisée	Non	Oui	Non
[Kasnesis et al., 2017]	Architecture serveur/objets	Non-défini	Centralisée	Non	Non	Non
[thingspeak, 2019] [Girau et al., 2013]	Architecture RESTful	C-WOR, SOR, C-LOR, OOR, POR	Centralisée	Non	Non	Non
[Girau et al., 2016]	Architecture PaaS	C-WOR, SOR, C-LOR, OOR, POR	Décentralisée	Oui	Non	Non
[Pintus et al., 2012]	Architecture publication/Souscription	Relations entre propriétaires d'objets	Distribuée	Oui	Non	Non
Notre approche	Architecture Fog/Cloud Computing	C-WOR, C-LOR et OOR	Distribuée	Oui	Oui	Oui

TABLE 3.1 – Synthèse des travaux de découverte IoT basés sur les réseaux sociaux

À travers cette étude, nous avons constaté que les approches de découverte proposées, soit pour les services Web traditionnels ou soit pour les services IoT, sont intéressantes vu qu'elles présentent les éléments et les concepts de bases nécessaires pour mettre en œuvre des applications distribuées. Néanmoins, ces approches souffrent de plusieurs limites. Tout d'abord, leur inconvénient majeur est qu'elles sont basées sur un gestionnaire central pour la gestion des relations sociales et la découverte des services. Par conséquent, s'il y a une défaillance, l'ensemble du système sera amputé de toutes ses fonctionnalités. En outre, ces approches centralisées sont limitées en terme de scalabilité. De plus, dans la plupart des approches proposées, diverses métriques sociales sont utilisées, cependant, elles ne reflètent pas vraiment les aspects sociaux nécessaires à la construction des réseaux sociaux, tels que : les centres d'intérêts et les capacités de collaboration. Ajoutant à cela, peu de détails sur les architectures, les modèles sociaux ainsi que les méthodes de découverte des services sont fournis dans leurs propositions.

Pour palier à ces limites, nous proposons une nouvelle approche de découverte des services IoT décentralisée basée sur un nouveau modèle social. Ce modèle s'appuie sur des mesures et des relations qui tiennent compte des aspects sociaux qui peuvent exister entre les objets IoT. En outre, il permet aux avatars de construire leurs réseaux sociaux d'une manière proactive et autonome pour accomplir des objectifs complexes. La décentralisation de notre approche de découverte sur un ensemble de réseaux sociaux la rend scalable et adaptée aux infrastructures IoT à large-échelle.

3.3 Contribution II : Découverte distribuée des services IoT basée sur les réseaux sociaux et le clustering

Le problème de découverte des services IoT fournis par les avatars peut être décrit comme suit : étant donné un processus abstrait qui modélise une application IoT (objectif) qu'un avatar, que nous appelons initiateur, ne peut pas accomplir tout seul, et un ensemble d'avatars disponibles, le but de notre approche est de permettre à l'initiateur de découvrir d'une manière distribuée un groupe d'avatars avec lesquels il peut collaborer pour achever son objectif.

Un objectif est associé à des règles réactives qui permettent de le déclencher. Une fois qu'un événement se produit, la règle dont les antécédents correspondent à l'événement produit est exécutée. Cette règle indique à l'avatar quel processus (application) doit être exécuté. Ensuite, l'avatar cherche la description sémantique du processus dans sa base de connaissances. Pour réaliser et exécuter ce processus, l'avatar déclenche une découverte distribuée d'avatars appropriés capables d'exécuter les actions qui composent le processus.

L'approche de découverte que nous proposons repose sur trois étapes principales, illustrées sur la Figure 3.1 : **a)** la construction du réseau social de l'initiateur qui

regroupe les avatars les plus susceptibles de participer à la réalisation de l'objectif, **b)** le clustering du réseau social pour classer les avatars selon leurs fonctionnalités afin d'orienter au mieux la transmission des requêtes de découverte aux avatars adéquats, et **c)** la propagation des requêtes de découverte dans les réseaux sociaux des avatars voisins. Ces étapes sont détaillées ci-dessous.

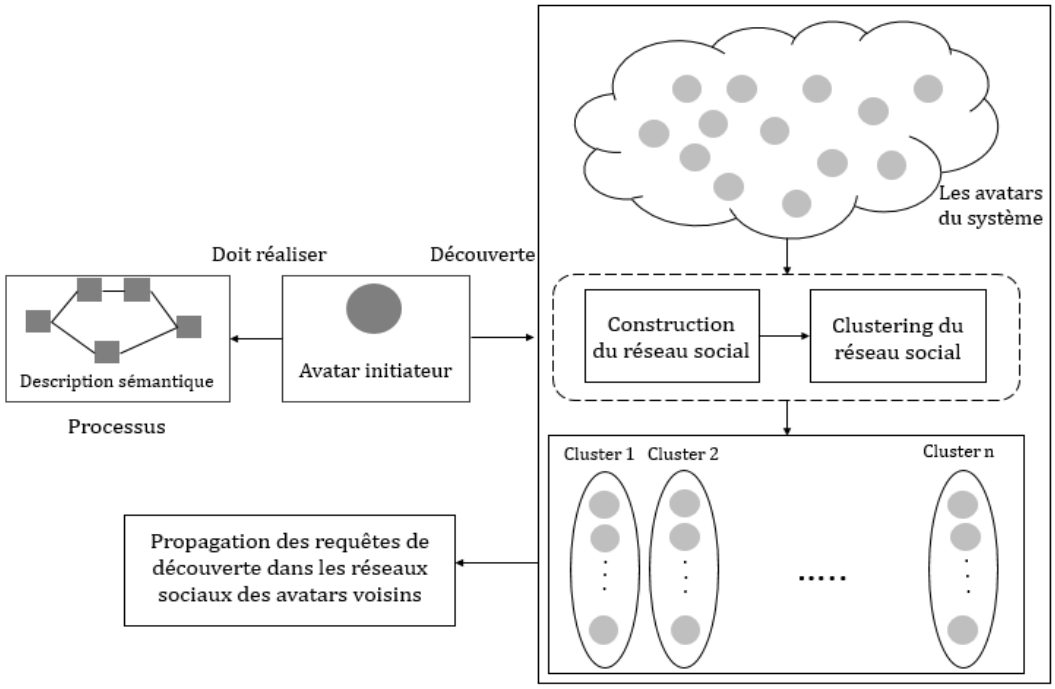


FIGURE 3.1 – Aperçu de l'approche de découverte sociale des services IoT

3.3.1 Construction du réseau social d'avatars

Les métadonnées (notamment : intérêts, position et utilisateur) des avatars du système IoT sont décrites sémantiquement et sont publiées dans des référentiels régionaux (nœuds fog et cloud). Avant de calculer les distances sociales avec les avatars du système, il est important de réduire le nombre d'avatars à considérer aux avatars qui ont au moins un intérêt commun pour minimiser les calculs. Pour cela, l'avatar initiateur exécute une requête SPARQL sur le référentiel régional correspondant auquel il est connecté. Cette requête est de la forme :

```
SELECT *
WHERE {
 ?resource avatarOnt:hasInterest ?Interest
 FILTER (?Interest IN
 (avatarOnt:I1,...,avatarOnt:Im))
}
```

Une fois que cette liste récupérée, l'avatar initiateur procède au calcul des distances sociales avec chaque avatar de cette liste. Pour ce faire, nous considérons en particulier les trois types de relations sociales : la relation de travail C-WOR, la relation de localisation C-LOR et la relation d'appartenance OOR. Le partage des mêmes sujets d'intérêt, de la localisation et du propriétaire ou groupe de propriétaires sont les trois axes sociaux les plus importants à considérer en IoT parmi les relations étudiées dans les travaux connexes.

3.3.1.1 Co-work Object Relationship (C-WOR)

Les avatars qui peuvent coopérer entre eux pour réaliser des tâches communes sont classés dans la classe C-WOR. Ces avatars possèdent au moins un intérêt en commun. Afin de mesurer quantitativement la similarité en terme de C-WOR entre deux avatars A_i et A_j, la similarité cosinus [Mei et al., 2011] est utilisée. Elle représente le rapport entre le produit scalaire de leurs vecteurs d'intérêts I_i et I_j et la norme de ces mêmes vecteurs. Cette similarité WS (Work Similarity), notée WS(A_i, A_j), est calculée comme indiqué en Équation 3.1 :

$$WS(A_i, A_j) = cos(I_i, I_j) = \frac{I_i.I_j}{\mid I_i \mid . \mid I_j \mid} \tag{3.1}$$

Où $\mid I_i \mid$ et $\mid I_j \mid$ sont les normes de I_i et I_j respectivement et $0 < WS(A_i, A_j) < 1$ de telle sorte que la valeur 0 indique que A_i et A_j sont indépendants et 1 indique qu'ils sont similaires et homologues.

3.3.1.2 Co-Location Objects Relationship (C-LOR)

Les avatars dont les objets IoT se trouvent dans la même zone géographique pendant un certain laps de temps ont tendance à collaborer ensemble dans le but de mener des objectifs en commun. La similarité entre deux avatars A_i et A_j d'un point de vue géographique est calculée en utilisant leurs coordonnées GPS (Global Positioning System) en se basant sur la distance à vol d'oiseau D donnée dans l'Équation 3.2 où R=6371Km est le rayon de la terre.

$$D(A_i, A_j) = R * acos[cos(LatA_i) * cos(LatA_j) * cos(LongA_j - LongA_i) + sin(LatA_i) * sin(LatA_j)] \quad (3.2)$$

La similarité géographique LS (Location Similarity), notée LS(A_i,A_j), est calculée comme indiqué dans l'Équation 3.3.

$$LS(A_i, A_j) = \frac{1}{D(A_i, A_j) + 1} \quad (3.3)$$

Cette fonction permet d'avoir une mesure $LS(A_i, A_j)$ élevée quand la distance géographique D est courte.

3.3.1.3 Ownership Object Relationship (OOR)

Cette relation sociale est utilisée lorsque des utilisateurs ou des sociétés préfèrent d'utiliser leurs propres objets IoT pour accomplir leurs applications IoT, par exemple pour des raisons de sécurité ou de confidentialité. La relation OOR, OS (Ownership Similarity), entre deux avatars A_i et A_j, notée OS(A_i,A_j), est donnée en Équation 3.4.

$$OS(A_i, A_j) = \begin{cases} 1 & \text{Si } A_i \text{ et } A_j \text{ appartient au même propriétaire ou même groupe} \\ 0 & \text{Sinon} \end{cases} \quad (3.4)$$

3.3.1.4 Similarité sociale globale

En se basant sur les distances de similarité des relations C-WOR, C-LOR et OOR détaillées ci-dessus, la similarité sociale globale, GS (Global Similarity) entre deux avatars A_i et A_j est calculée en utilisant une somme pondérée de ces trois similarités. Elle est modélisée par une fonction $GS(A_i, A_j)$ dans [0,1] comme indiquée dans l'Équation 3.5.

$$GS(A_i, A_j) = \alpha * WS(A_i, A_j) + \beta * LS(A_i, A_j) + \gamma * OS(A_i, A_j) \quad (3.5)$$

Les coefficients de pondération $\alpha, \beta, \gamma \in [0, 1]$ avec $\alpha+\beta+\gamma=1$ représentent les préférences pour chaque type de relation sociale.

3.3.1.5 Application : Découverte des services du processus de dépassement

Pour illustrer le processus de découverte que nous proposons, nous considérons l'exemple de dépassement fourni dans la section 2.2.3.4.

Les véhicules, la RSU, le GNSS et les différents objets connectés sont associés à leurs images virtuelles (avatars) et leurs descriptions sont publiées dans le serveur régional. Dans ce scénario, une fois que le clignotement à gauche soit activé, l'avatar A_1 qui représente le véhicule qui veut dépasser commence par raisonner sur sa base de connaissances pour connaître les tâches qu'il peut accomplir tout seul du processus de dépassement. Ce dernier constate qu'il peut effectuer les tâches : T_3 - mesurer la distance par rapport au véhicule devant, T_4 - mesurer la vitesse du véhicule devant, T_5 - calculer la différence de vitesse avec le véhicule devant, T_8 - calculer la distance requise pour un dépassement sécurisé et T_9 - comparer entre la distance de sécurité mesurée et celle calculée. Ainsi, l'avatar du véhicule hôte ne peut pas effectuer T_1, T_2, T_6, T_7 (traitement d'images perçues de l'environnement, mesure de distance et de vitesse des autres véhicules). Par conséquent, l'avatar A_1 commence à exécuter une approche de découverte basée sur le réseautage social et le clustering pour trouver les avatars qui peuvent accomplir T_1, T_2, T_6, T_7.

Pour cela, A_1 cherche tout d'abord à trouver les avatars qui ont au moins un intérêt en commun en exécutant la requête SPARQL suivante sur le serveur régional auquel l'avatar A_1 est connecté.

```
SELECT *
WHERE {
?resource avatarOnt:hasInterest ?Interest
FILTER (?Interest IN
(avatarOnt:I1, avatarOnt:I2, avatarOnt:I4))
}
```

Une fois que la liste des avatars trouvés est retournée, A_1 procède au calcul des distances sociales qui le sépare de chacun des avatars retournés par la requête SPARQL. Pour cela, nous fixons les préférences utilisateur comme suit : $\alpha = 0{,}4$, $\beta = 0{,}5$ et $\gamma = 0{,}1$ compte tenu de l'importance de la localisation dans un tel scénario. Le Tableau 3.2 donne les différentes distances sociales calculées.

Après avoir calculé les distances de similarité sociale, un seuil, appelé seuil de réseau social, est utilisé pour choisir les avatars qui feront partie du réseau social. Ce seuil est fixé à 0.6 ce qui permet de choisir des avatars socialement proches, tout en garantissant une taille acceptable du réseau social.

En prenant le seuil d'appartenance au réseau social de A_1 égal à 0,6 et en considérant les distances sociales calculées ci-dessous, nous constatons que le réseau social de A_1 englobe : A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9 et A_{12}.

Avatar	Similarité sociale	Avatar	Similarité sociale
A_2	0.8329	A_{10}	0.2865
A_3	0.8214	A_{11}	0.5454
A_4	0.8214	A_{12}	0.6910
A_5	0.7713	A_{13}	0.5880
A_6	0.6142	A_{14}	0.5758
A_7	0.6650	A_{15}	0.5869
A_8	0.8274	A_{16}	0.5880
A_9	0.6429	A_{17}	0.5758
A_{18}	0.5869		

TABLE 3.2 – Distances sociales par rapport à l'avatar du véhicule hôte A_1

3.3.2 Clustering du réseau social d'avatars

Après la construction de son réseau social, l'avatar initiateur procède à la répartition de ses avatars voisins selon leurs fonctionnalités pour rendre la découverte plus efficace en diminuant l'espace de recherche et orienter au mieux la transmission des requêtes de découverte aux avatars adéquats.

Le clustering [Grira et al., 2004] est défini comme un processus automatique qui permet de partitionner ou de regrouper un ensemble de données ou d'objets selon des critères de similitude en utilisant plusieurs types de distances. Il s'agit d'une méthode d'apprentissage non-supervisé et d'analyse statistique. Elle est utilisée pour construire des groupes de données homogènes, appelés clusters, à partir d'un grand ensemble de points de données mixtes et hétérogènes. Les clusters sont caractérisés par les propriétés suivantes : **a)** l'homogénéité intra-clusters, i.e., les points de données appartenant au même cluster présentent des caractéristiques similaires, et **a)** l'hétérogénéité inter-clusters, i.e., les points de données des différents clusters sont les plus dissemblables possibles. Il existe plusieurs catégories d'algorithmes de classification non-supervisée, principalement : les algorithmes k-means et les algorithmes fuzzy c-means.

L'algorithme k-means est introduit par McQueen en 1967 [MacQueen et al., 1967]. Il permet de répartir les points de données en plusieurs clusters, dont le nombre K est fixé à priori, d'une manière stricte, i.e., un point de données ne peut appartenir qu'à un seul cluster à la fois. Le principe est de définir K points de l'espace de données pour représenter les centroïdes initiaux des clusters à construire. La définition des centroïdes initiaux est importante, car cela influence considérablement le résultat de clustering final. Une fois que les centroïdes des clusters sont définis, chaque point de données est assigné au cluster dont le centroïde est le plus proche. Lorsque tous les points sont considérés, K nouveaux centroïdes sont déterminés. Ils sont recalculés en s'appuyant sur la moyenne des valeurs des points qui se trouvent au sein du même cluster. Ensuite, les étapes de calcul des centroïdes et d'assignation des points au cluster dont le centroïde

est le plus proche sont répétés jusqu'à ce que plus aucun changement ne soit effectué, i.e., les centroïdes ne changent plus.

Quant à l'algorithme fuzzy c-means, il est défini comme une méthode d'apprentissage non-supervisé dérivée de l'algorithme k-means décrit précédemment. Elle a été introduite par Besdek en 1981 [Bezdek, 1981]. Le principal avantage et la particularité de cette méthode est qu'elle introduit la notion de degré d'appartenance à plusieurs clusters, comme en logique floue, plutôt qu'une assignation stricte et exigée à l'un des clusters comme dans la méthode k-means. Cela signifie qu'un point de données peut appartenir à plusieurs clusters avec différents degrés d'appartenance. L'idée principale de cet algorithme est d'effectuer des regroupements via un processus itératif qui a pour objectif de minimiser une fonction objective avec la mise à jour des coefficients d'appartenance et des centroïdes des clusters qui sont étroitement liés. Formellement, pour chaque point de données x_i, i.e, le vecteur de fonctionnalités de l'avatar A_i, un coefficient, donnant le degré d'appartenance au $j^{ème}$ cluster, u_{ij} est défini. La somme de ces coefficients pour le même point x_i doit vérifier la condition suivante :

$$\sum_{j=1}^{C} u_{ij} = 1 \tag{3.6}$$

Tel que : C est le nombre de clusters à construire.

Dans cet algorithme, le centroïde c_j du $j^{ème}$ cluster représente la moyenne de tous les points de données pondérée par leur degré d'appartenance à ce même cluster. Il est donné par l'Équation 3.7.

$$c_j = \frac{\sum_{i=1}^{N} [u_{ij}]^m x_i}{\sum_{i=1}^{N} [u_{ij}]^m} \tag{3.7}$$

Tel que : N est le nombre de points de données considérés et m est le paramètre de fuzzification. Plus le paramètre de fuzzification m est élevé, plus les appartenances aux clusters sont lissées. Dans le cas où m=1, l'algorithme fuzzy c-means devient équivalent à k-means.

Le degré d'appartenance de chaque point x_i à chaque cluster j est mis à jour dans chaque itération de l'algorithme comme indiqué en Équation 3.8.

$$u_{ij} = \frac{1}{\sum_{z=1}^{C} \left(\frac{Dist(x_i, c_j)}{Dist(x_i, c_z)}\right)^{\frac{2}{m-1}}} \tag{3.8}$$

Tel que : $Dist$ est la distance utilisée pour mesurer la similitude entre les centroïdes et les points de données.

Nous avons choisi de nous appuyer sur l'algorithme fuzzy c-means pour partitionner les avatars voisins en plusieurs clusters selon les fonctionnalités qu'ils fournissent. Ce choix est justifié par le fait qu'un avatar peut fournir plusieurs fonctionnalités et par conséquent, il peut faire partie de plusieurs clusters. L'algorithme 1 présente la procé-

dure de clustering fuzzy c-means. Dans cet algorithme, chaque ligne de la matrice de données d'avatars représente le vecteur de fonctionnalités d'un avatar sous forme binaire où la composante vaut 1 si l'avatar peut effectuer la fonctionnalité correspondante et 0 sinon.

Algorithme 1 : Algorithme Fuzzy C-means des avatars

Entrées :
N : nombre d'avatars
$X = [x_{ij}]$: matrice de données d'avatars
C : nombre de clusters
Sorties : $U = [u_{ij}]$: matrice des degrés d'appartenance

Début

Étape 1. Initialiser d'une manière aléatoire la matrice U, $U^{(0)}$, tel que les degrés d'appartenance des avatars de même cluster C_j vérifient :

$$\sum_{j=1}^{C} u_{ij} = 1$$

Étape 2. Calculer le centroïde c_j de chaque cluster j, tel que :

$$c_j = \frac{\sum_{i=1}^{N} [u_{ij}]^m x_i}{\sum_{i=1}^{N} [u_{ij}]^m}$$

Étape 3. Mettre à jour la matrice $U^{(k)}$, et construire $\mathrm{U}^{(k+1)}$, tel que :

$$u_{ij} = \frac{1}{\sum_{z=1}^{C} (\frac{Dist(x_i, c_j)}{Dist(x_i, c_z)})^{\frac{2}{m-1}}}$$

si $(\sum_{i=1}^{N} \sum_{j=1}^{C} [u_{ij}]^m Dist(x_i, c_j)^2 < \varepsilon)$ **alors**
    Fin du clustering
fin
sinon
    Retourner à Étape 2.
fin
Fin

Après la répartition des avatars voisins selon leurs fonctionnalités en plusieurs clusters, un avatar est désigné comme un délégué dans chaque cluster. Il représente l'avatar avec le plus grand degré d'appartenance. Le rôle de cet avatar délégué est de gérer les requêtes envoyées à son cluster pour découvrir les avatars appropriés répondant aux besoins de l'application. L'objectif est de garantir une découverte distribuée et une propagation utile des requêtes de découverte en minimisant au plus la redondance des messages échangés entre les avatars.

3.3.2.1 Application : clustering du réseau social dans le scénario de dépassement

Après l'étape de la construction de son réseau social, l'avatar de véhicule hôte A_1 procède à la répartition de ce réseau en plusieurs clusters selon les fonctionnalités que les avatars fournissent. Donc, en appliquant l'algorithme fuzzy c-means sur une matrice composée des fonctionnalités fournies par les avatars de notre scénario, nous obtenons les résultats suivants :

- Le premier cluster C_1 qui représente la fonctionnalité de traitement d'images perçues de l'environnement, il contient les avatars suivants ordonnés selon leur degré d'appartenance : A_4, A_2, A_8, A_{10}, A_{16}, A_{15}, A_{13}, A_{18} et A_5.
- Le deuxième cluster C_2 qui représente la mesure de distance, il contient les avatars suivants ordonnés selon leur degré d'appartenance : A_7, A_8, A_{10}, A_{17}, A_{16}, A_{12} et A_{15}.
- Le troisième cluster C_3 représente la mesure de vitesse, il contient les avatars suivants ordonnés selon leur degré d'appartenance : A_3, A_7, A_4, A_2, A_{10}, A_{13}, A_{17}, A_{16} et A_{14}.
- Le quatrième cluster C_4 représente la fonctionnalité de calcul, il contient les avatars suivants ordonnés selon leur degré d'appartenance : A_2, A_3, A_4, A_{12} et A_{13}.

Le premier avatar de la liste des avatars de chaque cluster est celui qui a le plus grand degré d'appartenance, i.e., qui a la distance la plus proche du centre de gravité du cluster, il sera désigné comme l'élu du cluster correspondant. Ainsi, A_4 sera l'élu du C_1, A_7 sera l'élu du C_2, A_3 sera l'élu du C_3 et A_2 sera l'élu du C_4.

3.3.3 Propagation des requêtes de découverte

Après la désignation des avatars élus, l'avatar initiateur procède à la découverte des avatars qui fournissent les services requis. Pour ce faire, pour chaque tâche du processus abstrait, il envoie une demande à l'avatar élu responsable de la fonctionnalité exprimée dans la tâche. Chaque élu se charge de rechercher le service demandé dans son cluster. Dès qu'un service est trouvé, il sera ajouté à l'ensemble des avatars candidats pour accomplir la tâche abstraite correspondante. Si aucun service n'est trouvé au niveau du

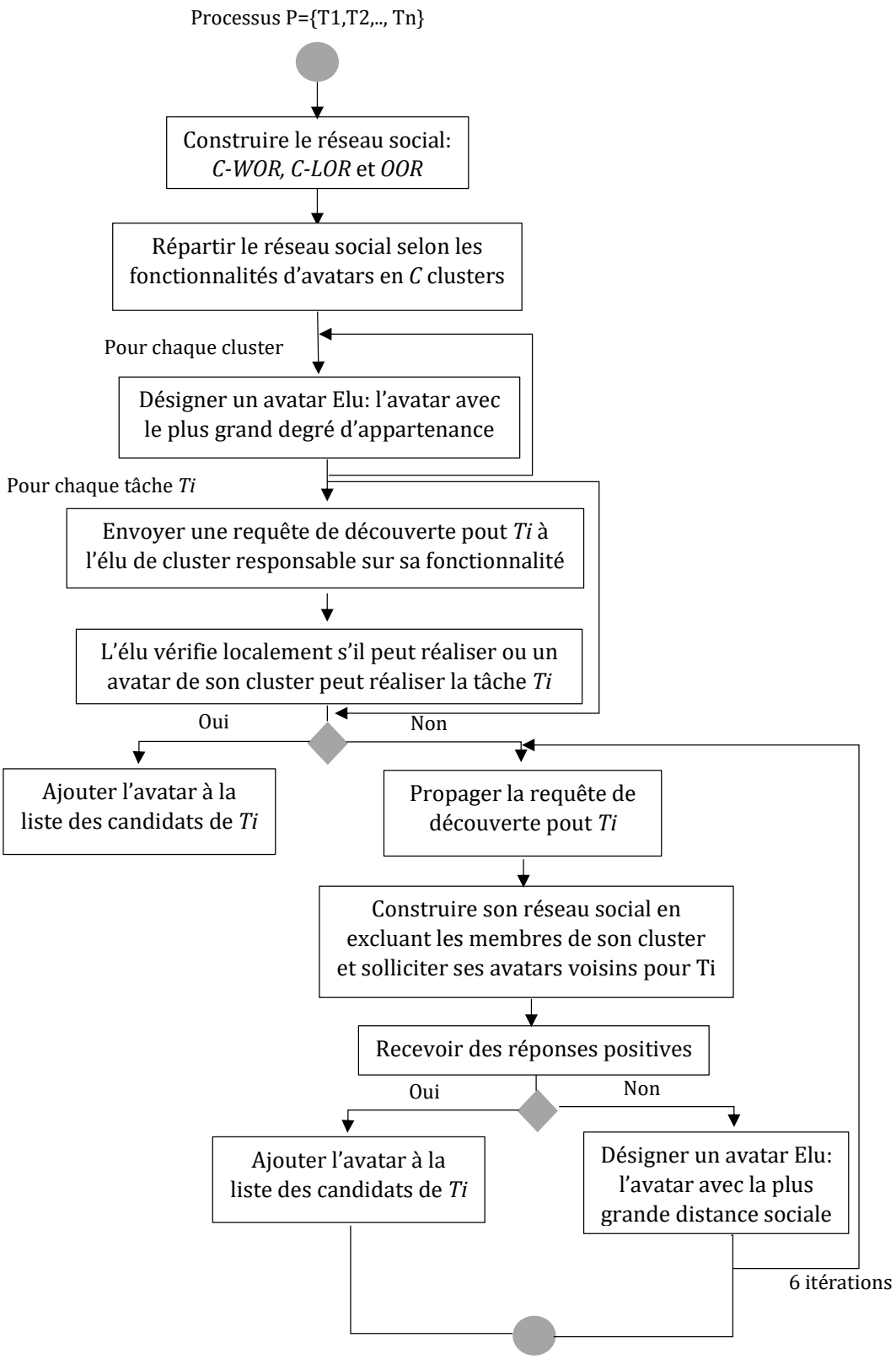
cluster, l'avatar élu propage la demande en créant son propre réseau social et désigne un avatar élu dans le nouveau réseau social pour effectuer la recherche. La propagation peut se poursuivre jusqu'au sixième niveau, en s'inspirant de l'effet du petit monde des réseaux sociaux en sociologie [Travers and Milgram, 2006]. Le résultat de cette étape est un ensemble de services d'avatars candidats pour chaque tâche du processus, ces ensembles peuvent être également appelés clusters.

3.3.3.1 Application : Découverte des services pour le scénario de dépassement

Pour découvrir les avatars qui peuvent satisfaire les tâches T_1 et T_2 qui correspondent à la fonctionnalité de traitement d'images perçues, l'avatar A_1 envoie une demande à l'élu A_4 qui s'occupe de trouver les services répondant à T_1 et T_2 dans le cluster C_1. Il constate que T_1 peut effectuer par lui-même ainsi que par les avatars A_{10}, A_8, A_{16}, A_{15} et A_{18}, et la tâche T_2 par A_2, A_{10}, A_8, A_{16}, A_{15} et A_{13}. De même, A_1 envoie une requête de découverte à l'élu A_7 du cluster C_2 concernant les tâches T_3 et T_6 qui correspondent à la fonctionnalité de mesure de distance. Après avoir solliciter les avatars de son cluster, A_7 déduit que la tâche T_3 peut être effectuée par lui-même et par les avatars A_8, A_{10}, A_{12} et A_{16}, et la tâche T_6 par les avatars A_8, A_{10}, A_7, A_{16} et A_{17}. En ce qui concerne les tâches T_4 et T_7 relatives à la fonctionnalité de mesure de vitesse, l'avatar A_1 envoie une requête de découverte à l'élu A_3 du cluster C_3. Les résultats de la découverte montre que T_4 peut être effectuée par A_2, A_7, A_{10}, A_{13} et A_{17}, et la tâche T_7 par A_4, A_7, A_{10}, A_{15}, A_{16} et A_{14}. Pour les autres tâches T_5, T_8 et T_9 qui font référence à une fonctionnalité de calcul, l'avatar A_1 sollicite l'élu A_2 du cluster C_4. L'élu A_2 trouve que ces tâches peuvent être effectuées par tous les avatars du cluster A_2, A_3, A_4, A_{12} et A_{13}.

Chaque élu insère dans la description sémantique du processus de dépassement plusieurs triplets pour désigner les avatars découverts pour chaque tâche abstraite et A_1 sera responsable de rassembler toutes ces informations dans une seule description sémantique et pour ensuite la partager avec les avatars participants à la collaboration comme l'indique l'illustration A.3 en Annexe A.

Pour récapituler, les différentes étapes du processus de découverte des services IoT fournis par les avatars, présentées précédemment, sont illustrées dans le diagramme global suivant :


Processus P={T1,T2,.., Tn}
Construire le réseau social: C-WOR, C-LOR et OOR
Répartir le réseau social selon les fonctionnalités d'avatars en C clusters
Pour chaque cluster
Désigner un avatar Elu: l'avatar avec le plus grand degré d'appartenance
Pour chaque tâche Ti
Envoyer une requête de découverte pout Ti à l'élu de cluster responsable sur sa fonctionnalité
L'élu vérifie localement s'il peut réaliser ou un avatar de son cluster peut réaliser la tâche Ti
Oui
Non
Ajouter l'avatar à la liste des candidats de Ti
Propager la requête de découverte pout Ti
Construire son réseau social en excluant les membres de son cluster et solliciter ses avatars voisins pour Ti
Recevoir des réponses positives
Oui
Non
Ajouter l'avatar à la liste des candidats de Ti
Désigner un avatar Elu: l'avatar avec la plus grande distance sociale
6 itérations

3.4 Conclusion

Dans ce chapitre, nous avons présenté notre deuxième contribution qui consiste en une nouvelle approche distribuée basée sur le réseautage social et les algorithmes de clustering pour la découverte des services IoT. Cette approche permet, d'une part, de réduire l'espace de recherche, et d'autre part, de mieux cibler la transmission des requêtes de services. La solution proposée est particulièrement intéressante dans le contexte de systèmes complexes avec un grand nombre d'objets hétérogènes.

Avant de détailler cette contribution, nous avons fourni tout d'abord un aperçu générique sur les réseaux sociaux et un état de l'art sur les approches existantes pour la découverte des services basée sur les réseaux sociaux. Après cela, nous avons présenté le modèle social construit et une vue d'ensemble du système sur lesquels notre approche de découverte est fondée. Ensuite, nous avons détaillé les étapes qui permettent à un avatar de construire son réseau social en se basant sur plusieurs types de relations sociales notamment la localisation et les intérêts. Puis, l'étape de clustering qui permet de partitionner le réseau social construit selon les fonctionnalités que les avatars offrent afin de guider la découverte des services IoT. Notre approche de découverte est évaluée dans la section 5.2.

Les applications IoT peuvent être soumises à des contraintes non-fonctionnelles. Nous nous intéressons particulièrement aux paramètres de qualité de services (Quality of Service, QoS). Aussi, les services IoT découverts sont généralement dotés de paramètres de QoS différentes. La sélection des avatars doit tenir compte de ces paramètres afin de satisfaire les applications IoT contraintes.

Dans ce contexte, nous proposons dans le chapitre suivant deux approches de sélection basées sur les paramètres de QoS : une approche basée sur la décomposition des contraintes de QoS globales en contraintes locales via un algorithme génétique évolutionnaire multi-objectifs (MOEA) réalisé via la collaboration avec les élus des clusters, et une approche basée sur la prédiction des contraintes de QoS locales via la méthode de régression SVM qui surdépasse la première en s'affranchissant l'étape de collaboration.

Chapitre 4

Vers une sélection distribuée basée sur le MOEA et la SVM

Sommaire

L'approche de découverte présentée précédemment permet de construire des clusters d'avatars qui présentent les mêmes fonctionnalités, mais qui se distinguent les uns des autres par leurs propriétés non-fonctionnelles (QoS). La Figure 4.1 donne une vue d'ensemble du système après l'étape de découverte.

Dans ce contexte distribué où la gestion centrale de la QoS présente des limites et où les applications sont soumises à des contraintes de QoS globales, nous proposons une approche de sélection distribuée qui vise à maximiser l'utilité et la fiabilité de la composition résultante tout en garantissant les exigences de QoS globales. Étant donné que les paramètres de QoS peuvent être fluctuants (i.e., ils peuvent varier dans le temps), l'approche que nous proposons tient en compte la fluctuation des paramètres de QoS.

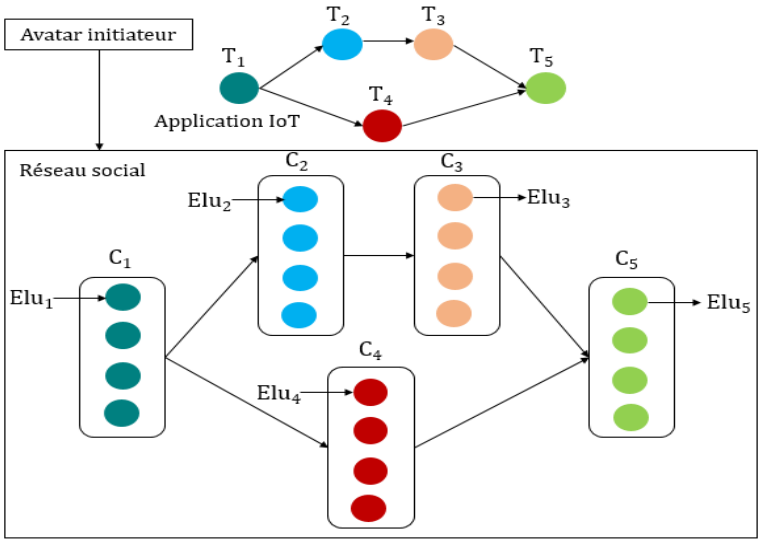


FIGURE 4.1 – Vue d'ensemble du système après l'étape de découverte

4.1 Problème de sélection locale

Avant de présenter les approches proposées, nous reprenons l'exemple de dépassement présenté dans l'inroduction pour illustrer le problème de sélection auquel nous nous intéressons.
Nous supposons que le processus de dépassement, introduit dans la section 2.2.3.4 est soumis à deux contraintes de QoS globales : le temps de réponse qui doit être inférieur à 400 ms (0.4s) et le débit qui doit être supérieur à 7 r/s (requêtes/seconde), avec des préférences (poids des attributs de QoS) : 0,6 pour le temps de réponse et 0,4 pour le débit. Le Tableau 4.1 regroupe pour chaque tâche le cluster des avatars découverts.

Tâche	Avatars candidats	Tâche	Avatars candidats
T_1	A_4, A_{10}, A_8, A_{16}, A_{15}, A_{18}	T_6	A_8, A_{10}, A_7, A_{16}, A_{17}, A_{15}
T_2	A_2, A_{10}, A_8, A_{16}, A_{15}, A_{13}	T_7	A_4, A_7, A_{10}, A_{15}, A_{16}, A_{14}
T_3	A_1, A_7, A_8, A_{10}, A_{16}, A_{12}	T_8	A_1, A_2, A_3, A_4, A_{12}, A_{13}
T_4	A_1, A_2, A_7, A_{10}, A_{13}, A_{17}	T_9	A_1, A_2, A_3, A_4, A_{12}, A_{13}
T_5	A_1, A_2, A_3, A_4, A_{12}, A_{13}		

TABLE 4.1 – Avatars candidats découverts pour chaque tâche

Il existe plusieurs instances de caméras, de radars, de lidars, de véhicules, et de piétons avec différents paramètres de QoS. Nous donnons en Tableau A.1 de l'Annexe A les informations autour des propriétés de QoS en termes de temps de réponse et de débit (throughput) et leurs dernières six valeurs historiques pour les avatars découverts. Par

exemple, les valeurs de temps de réponse de l'avatar A_1 sont : 8, 85, 69, 7, 21 et 72, et les valeurs de débit sont : 21, 3, 11, 24, 11 et 26.

L'objectif est de trouver une combinaison quasi-optimale d'avatars sans avoir à tester toutes les combinaisons possibles et de permettre de sélectionner chaque avatar indépendamment des autres avatars.

Pour cela, nous proposons d'abord une approche de sélection génétique évolutionnaire multi-objectifs (Multi-Objective Evolutionary Algorithm, MOEA) basée sur la décomposition des contraintes de QoS globales en contraintes locales. Ceci permet d'effectuer une sélection des avatars basée sur une sélection locale. Cette approche est ensuite étendue pour prédire les contraintes de QoS locales en appliquant la méthode de régression SVM.

4.2 Notions préliminaires

Le Tableau 4.2 donne l'ensemble des notions préliminaires utilisées par la suite dans ce chapitre pour nos approches de sélection.

Notion	Modélisation
Processus (Application IoT)	$P = \{T_1, T_2, \ldots, T_n\}$, tq : n est le nombre de tâches abstraites
Cluster	$C_j = \{A_{1j}, A_{2j}, \ldots, A_{lj}\}$, tq : A_{ij} est le $i^{ème}$ avatar du $j^{ème}$ cluster et l est le nombre d'avatars dans C_j
Vecteur QoS d'un avatar A_{ij}	$Q_{ij} = \left\{q_{ij}^1, q_{ij}^2, \ldots, q_{ij}^r\right\}$, tq : q_{ij}^k est la valeur de $k^{ème}$ attribut de QoS de A_{ij} et r est le nombre d'attributs de QoS
Poids de préférences	$W = \{w_1, w_2, \ldots, w_r\}$, tq : $w_k (1 \leq k \leq r)$ est la poids donné au $k^{ème}$ paramètre de QoS, $w_i \in [0, 1]$ et $\sum_{i=1}^{r} w_i = 1$
Contraintes de QoS globales	$E = \{E_1, E_2, \ldots, E_r\}$, tq : $E_k (1 \leq k \leq r)$ est la contrainte de QoS globale par rapport au $k^{ème}$ paramètre de QoS
Contraintes de QoS locales	$\{e_{1k}, e_{2k}, \ldots, e_{nk}\}$ représentent les n contraintes locales obtenues après décomposition de la contrainte globale E_k
Niveaux de qualité	$QL_{jk} = \left\{L_{jk}^1, L_{jk}^2, \ldots, L_{jk}^d\right\}$, tq : L_{jk}^z est le $z^{ème}$ niveau de qualité pour le $k^{ème}$ attribut de QoS dans le $j^{ème}$ cluster
Min et Max de QoS	$max(q_j^k)$ est la valeur maximale du $k^{ème}$ paramètre de QoS dans son cluster C_j et $min(q_j^k)$ est sa valeur minimale

TABLE 4.2 – Notions et modélisation

4.2.1 Modèles de composition de QoS

La valeur de QoS globale d'un processus donné dépend de deux facteurs principaux : ses attributs de QoS et les structures qui interconnectent les tâches qui le composent. Notre approche considère les modèles couramment utilisés par les langages de composition comme BPEL [Juric et al., 2006] : séquentiel, parallèle, conditionnel et boucle. Chacune de ces structures peut être transformée en un modèle séquentiel en utilisant les techniques proposées dans [Wang et al., 2011]. Bien que notre approche soit générique en terme de nombre de paramètres de QoS, dans notre cas d'application, nous considérerons uniquement deux paramètres quantitatifs de QoS :

- Temps de réponse (Rt) : temps d'exécution moyen entre l'envoi de la demande et la réception de sa réponse.
- Débit (Throughput, Th) : nombre total de requêtes exécutées dans une période donnée. Ce paramètre reflète également la disponibilité du service.

Le Tableau 4.3 donne la fonction d'agrégation (Agg) de chaque attribut de QoS pour chaque structure de liaison comme mentionnée ci-dessus.

	Séquentiel	Parallèle	conditionnel	Boucle
Rt (q^1)	$\sum_{j=1}^{n} Rt_j$	$\max_{j=1}^{n} Rt_j$	$\max_{j=1}^{n} Rt_j$	$\sum_{j=1}^{k} Rt_j$
Th (q^2)	$\min_{j=1}^{n} Th_j$	$\min_{j=1}^{n} Th_j$	$\min_{j=1}^{n} Th_j$	Th_j

TABLE 4.3 – Fonctions d'agrégation de QoS

4.2.2 Utilité QoS

L'utilité QoS est une mesure qui représente le score à attribuer à un avatar donné pour mesurer le profit de le sélectionner par rapport à d'autres. Vu qu'un avatar possède plusieurs paramètres de QoS, nous avons choisi d'utiliser la technique SAW (Simple Additive Weighting) [Afshari et al., 2010] pour évaluer son utilité de QoS multidimensionnelle. Cette technique permet de mapper le vecteur de QoS de l'avatar à une seule valeur tout en tenant compte des préférences de l'application. Les attributs de QoS ont différentes unités et des intervalles de variation différents. Une normalisation doit être effectuée pour obtenir des mesures uniformes dans [0,1] avant le calcul de l'utilité. Elle dépend également du type des attributs de QoS. Ces derniers peuvent être classés comme positifs/négatifs ou statiques/dynamiques. Les paramètres positifs se réfèrent aux attributs à maximiser, tandis que les négatifs sont à minimiser. Les paramètres statiques ont une valeur fixe et ne varient pas dans le temps, alors que les attributs dynamiques sont de nature opposée.

La formule de normalisation est donnée par l'Équation 4.1 :

$$q_{ij}^k = \begin{cases} \frac{max(q_j^k) - q_{ij}^k}{max(q_j^k) - min(q_j^k)} & \text{Pour les paramètres négatifs} \\ \frac{q_{ij}^k - min(q_j^k)}{max(q_j^k) - min(q_j^k)} & \text{Pour les paramètres positifs} \\ 1 & max(q_j^k) - min(q_j^k) = 0 \end{cases} \tag{4.1}$$

Après normalisation de tous les paramètres de QoS, la fonction d'utilité de l'avatar A_{ij} est donnée en Équation 4.2 :

$$U(A_{ij}) = \sum_{k=1}^{r} w_k . q_{ij}^k \tag{4.2}$$

La valeur d'utilité d'une composition de n avatars AC est formulée dans l'Équation 4.3, tel que : A_{sj} est l'avatar sélectionné du cluster C_j.

$$U(AC) = \sum_{j=1}^{n} U(A_{sj}) \tag{4.3}$$

4.2.3 Fluctuation QoS

La fluctuation d'un paramètre de QoS est définie comme la distribution de ses valeurs au cours du temps. Les services dont la fluctuation des paramètres de QoS est élevée, ne sont pas souhaitables pour faire partie de la sélection finale, car leurs valeurs de QoS peuvent diverger au moment de l'exécution. Le concept de fluctuation est un facteur très important qui maximise la fiabilité de la composition finale. Il est calculé en s'appuyant sur le coefficient de variation (CV) qui représente le rapport entre l'écart-type et la moyenne des valeurs historiques des paramètres de QoS.

Soit q_{ij}^k le $k^{ème}$ paramètre de QoS de l'avatar A_{ij} et $\left\{q_{ij}^{k1}, q_{ij}^{k2}, \ldots, q_{ij}^{kh}\right\}$ les dernières h valeurs historiques de QoS de qij^k. h est généralement fixé à 6 ce qui permet d'observer suffisamment la variation des paramètres de QoS.

$$CV(q_{ij}^k) = \frac{Sd(q_{ij}^k)}{Av(q_{ij}^k)} \tag{4.4}$$

Tels que :
$Sd(q_{ij}^k) = \sqrt{\frac{1}{h-1}\sum_{z=1}^{h}(q_{ij}^{kz} - Av(q_{ij}^k))^2}$ représente l'écart type de q_{ij}^k. $Av(q_{ij}^k) = \frac{1}{h}\sum_{z=1}^{h} q_{ij}^{kz}$ est la moyenne des valeurs historiques de q_{ij}^k.

La fluctuation multidimensionnelle pour les r attributs de QoS de l'avatar A_{ij} est calculée comme indiqué en Équation 4.5 :

$$F(A_{ij}) = \sum_{k=1}^{r} w_k \frac{max(CV(q_{ij}^k)) - CV(q_{ij}^k)}{max(CV(q_{ij}^k)) - min(CV(q_{ij}^k))} \tag{4.5}$$

La valeur de fluctuation d'une composition de n avatars AC est formulée dans l'Équation 4.6, tel que : A_{sj} est l'avatar sélectionné du cluster C_j.

$$F(AC) = \sum_{j=1}^{n} F(A_{sj}) \tag{4.6}$$

4.3 Décomposition des contraintes de QoS globales

Le problème de sélection de services, soumis à des contraintes de QoS globales, peut être modélisé comme un problème d'optimisation combinatoire dont la fonction objective vise à maximiser la valeur d'utilité globale de la composition finale et à minimiser sa fluctuation tout en respectant les contraintes de QoS globales. Cependant, cela peut présenter des limites, notamment à cause du problème de l'explosion combinatoire pour des systèmes complexes.
Étant donné un processus abstrait P composé de n tâches abstraites avec r contraintes de QoS globales, le but de notre approche est de décomposer chaque contrainte de QoS globale E_k du $k^{ème}$ attribut de QoS en n contraintes locales $\{e_{1k}, e_{2k}, \ldots, e_{nk}\}$ affectées à chaque tâche. Ces contraintes sont utilisées comme bornes supérieures/inférieures pour la sélection locale distribuée et parallèle d'un avatar pour chaque tâche abstraite d'une manière indépendante, tels que :

- L'utilité globale $U(AC)$ est maximisée.
- La fluctuation globale $F(AC)$ est minimisée.
- Les contraintes de QoS globales sont satisfaites.

La Figure 4.2 donne un aperçu global de la solution de sélection distribuée que nous proposons.

Elle est basée sur la décomposition des contraintes de QoS de bout-en-bout (i.e., globales du processus) en contraintes locales en tenant compte de la fluctuation de QoS. Elle est réalisée via trois étapes principales : **1)** le calcul des niveaux de qualité pour déterminer les valeurs de QoS candidates pour être des contraintes locales, et cela, pour chaque attribut de QoS et chaque cluster (chaque élu se charge de calculer les niveaux de qualité de son cluster), **2)** la résolution du problème de décomposition des contraintes de QoS globales basée sur un algorithme générique évolutif multi-objectifs pour déterminer les contraintes locales parmi les niveaux de qualité calculés précédemment, et **3)** la sélection locale pour chaque tâche du processus basée sur les contraintes locales obtenues comme bornes supérieures/inférieures pour sélectionner l'avatar adéquat de chaque cluster d'une manière parallèle et indépendante.

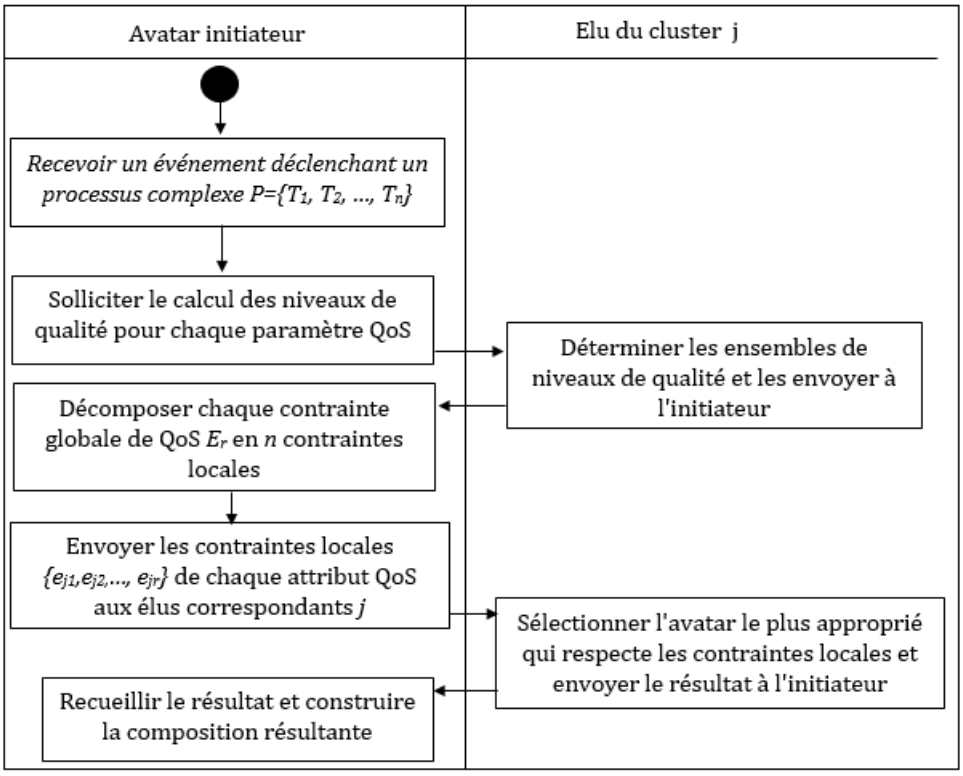


FIGURE 4.2 – Aperçu de la sélection basée sur la décomposition des contraintes de QoS globales

4.3.1 Détermination des niveaux de qualité

L'étape de calcul des niveaux de qualité vise à déterminer les contraintes de QoS locales candidates parmi les valeurs de QoS des avatars présents dans chaque cluster. Pour cela, la plage des valeurs de chaque paramètre de QoS est divisée en d valeurs discrètes appelées *niveaux de qualité* $QL_{jk} = \left\{L^1_{jk}, L^2_{jk}, \ldots, L^d_{jk}\right\}$, tels que : L^z_{jk} est le $z^{ème}$ niveau de qualité pour le $k^{ème}$ attribut de QoS dans le $j^{ème}$ cluster, $min(q^k_j) \leq L^1_{jk} \leq L^2_{jk} \leq \ldots \leq L^d_{jk} \leq max(q^k_j)$.

Pour le calcul des niveaux de qualité, nous nous sommes inspirés de l'algorithme proposé dans [Alrifai et al., 2012]. Le principe est le suivant : pour chaque attribut de QoS et pour chaque cluster, l'algorithme commence par trier les avatars candidats de C_j selon leurs valeurs de QoS pour le $k^{ème}$ attribut. Ensuite, ces avatars sont divisés en d sous-ensembles. Après cela, un avatar est choisi au hasard dans chaque sous-ensemble, et sa valeur de QoS de $k^{ème}$ attribut est ajoutée en tant que niveau de qualité à l'ensemble QL_{jk}. Cette méthode est donnée dans l'algorithme 2.

Algorithme 2 : Calcul des niveaux de qualité

Entrées :
n clusters,
l : le nombre d'avatars dans chaque cluster,
d : le nombre de niveaux de qualité tel que : $d \leq m$,
r contraintes de QoS globales
Sorties : r ensembles de niveaux de qualité

```
1  Début
2  pour chaque attribut de QoS q^k faire
3      pour chaque cluster C_j faire
4          QL_jk ← ∅ C_j ← Sort(C_j, q^k) Diviser C_j en d sous-ensembles :
           {C^1_jk, C^2_jk, ..., C^d_jk}
5          pour z ← 1 à d faire
6              L^z_jk ← SélectionAléatoire(C^z_jk)
7              QL_jk ← QL_jk ∪ {L^z_jk}
8          fin
9      fin
10 fin
11 Fin
```

La sélection d'un niveau de qualité L^z_{jk} comme contrainte locale parmi plusieurs autres dépend de l'évaluation de deux facteurs : l'utilité et la fluctuation.

A. Évaluation de l'utilité d'un niveau de qualité L^z_{jk}

Pour évaluer l'utilité d'un niveau de qualité L^z_{jk}, une fonction fitness p^z_{jk} dans [0,1] est utilisée. Elle représente la probabilité avec laquelle il serait intéressant d'utiliser L^z_{jk} comme contrainte locale. La formule est donnée en Équation 4.7 :

$$p^z_{jk} = \frac{h(L^z_{jk})}{l} \cdot \frac{u(L^z_{jk})}{u_{max}} \tag{4.7}$$

- $h(L^z_{jk})$ indique le nombre d'avatars qualifiés lorsque L^z_{ik} est sélectionné comme contrainte locale.
- l représente le nombre total d'avatars dans le cluster C_j.
- $u(L^z_{jk})$ est la plus grande utilité lorsque seuls les avatars qualifiés pour L^z_{ik} sont pris en compte.
- u_{max} est la plus grande utilité quand tous les avatars de C_j sont considérés.

B. Évaluation de la fluctuation d'un niveau de qualité L^z_{jk}

Pour évaluer la fluctuation d'un niveau de qualité L^z_{ik}, une fonction de fluctuation

f^z_{jk} dans [0,1] est également utilisée. Cette fonction reflète l'inconvénient si L^z_{ik} est sélectionné comme contrainte locale comme donné en Équation 4.8 :

$$f^z_{jk} = \frac{h(L^z_{jk})}{l} \cdot \frac{f_{min}}{f(L^z_{jk})} \tag{4.8}$$

- $h(L^z_{jk})$ indique le nombre d'avatars qualifiés lorsque L^z_{ik} est sélectionné comme contrainte locale.
- l représente le nombre total d'avatars dans le cluster C_j.
- f_{min} est la fluctuation la plus faible obtenue en considérant tous les avatars de C_j.
- $f(L^z_{jk})$ indique la fluctuation la plus faible obtenue lorsque seuls les avatars qualifiés pour L^z_{ik} sont pris en compte.

Application : Calcul des niveaux de qualité dans le scénario de dépassement

Pour le calcul des niveaux de qualité, nous commençons par détailler cette étape pour le premier cluster C_1, et cela, pour le temps de réponse en fixant le nombre de niveaux de qualité d à 3, comme le montre la Figure 4.3.

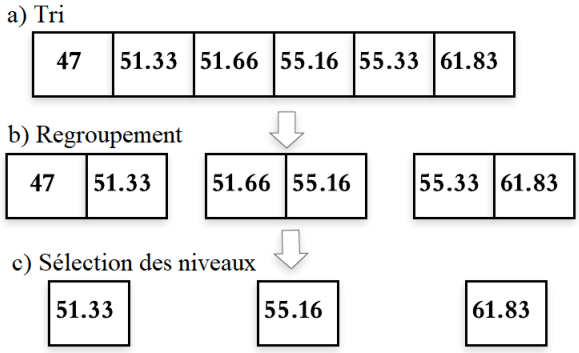


FIGURE 4.3 – Détermination des niveaux de qualité pour le processus de dépassement

Dans le cluster C_1, il y a 6 avatars candidats. Ainsi, pour déterminer les niveaux de qualité du temps de réponse : a) nous trions ces avatars selon les valeurs de leur temps de réponse. Après cela, b) l'ensemble des avatars triés est divisé en 3 groupes. Ensuite, c) nous sélectionnons un avatar au hasard dans chaque sous-groupe. Sa valeur sera rajoutée à la liste des niveaux de qualité QL_{11}. Par conséquent, la liste des niveaux de qualité obtenue est : $QL_{11} = \{51.33, 55.16, 61.83\}$.

De la même manière, nous obtenons les niveaux de qualité pour les autres clusters comme illustré dans le Tableau 4.4.

Cluster	Temps de réponse	Débit
C_1	51.33, 55.16, 61.83	10.66, 14.5, 18.83
C_2	39.5, 45.33, 64.66	12, 12.83, 21.33
C_3	29.33, 58.5, 61.16	10.33, 12.66, 16.5
C_4	45.33, 46.66, 47.5	9.66, 16.16, 19
C_5	43.66, 50.16, 60	11, 14.33, 16
C_6	36.33, 50, 50.5	10, 14.83, 18.5
C_7	46.66, 49.5, 61	10, 16.16, 19
C_8	52.16, 53.16, 71.33	11.83, 12, 16.66
C_9	28, 40.66, 69.66	14.83, 18, 18.16

TABLE 4.4 – Les niveaux de qualité calculés

Une fois que les niveaux de qualité sont calculés, nous procédons à leur évaluation en termes de temps de réponse et de débit. Le Tableau A.2 et le Tableau A.3 en Annexe A donnent l'évaluation des niveaux de qualité selon le temps de réponse et le débit respectivement.

4.3.2 Décomposition des contraintes de QoS globales via MOEA

Dans cette section, nous recherchons une combinaison de niveaux de qualité pour chaque cluster pour être utilisées comme contraintes de QoS locales. Chaque paramètre de QoS possède plusieurs niveaux de qualité, et cela, pour chaque cluster, et par conséquent, plusieurs schémas de niveaux de qualité sont possibles. Ce problème vise à optimiser simultanément l'utilité et la fluctuation de QoS des niveaux de qualité choisis sous un ensemble de contraintes de QoS globales. Il s'agit donc d'un problème d'optimisation combinatoire multi-objectifs (bi-objectifs). Ce dernier est formulé comme indiqué en Équation 4.9, tel que : les formules d'agrégation Agg sont déjà données dans le Tableau 4.3.

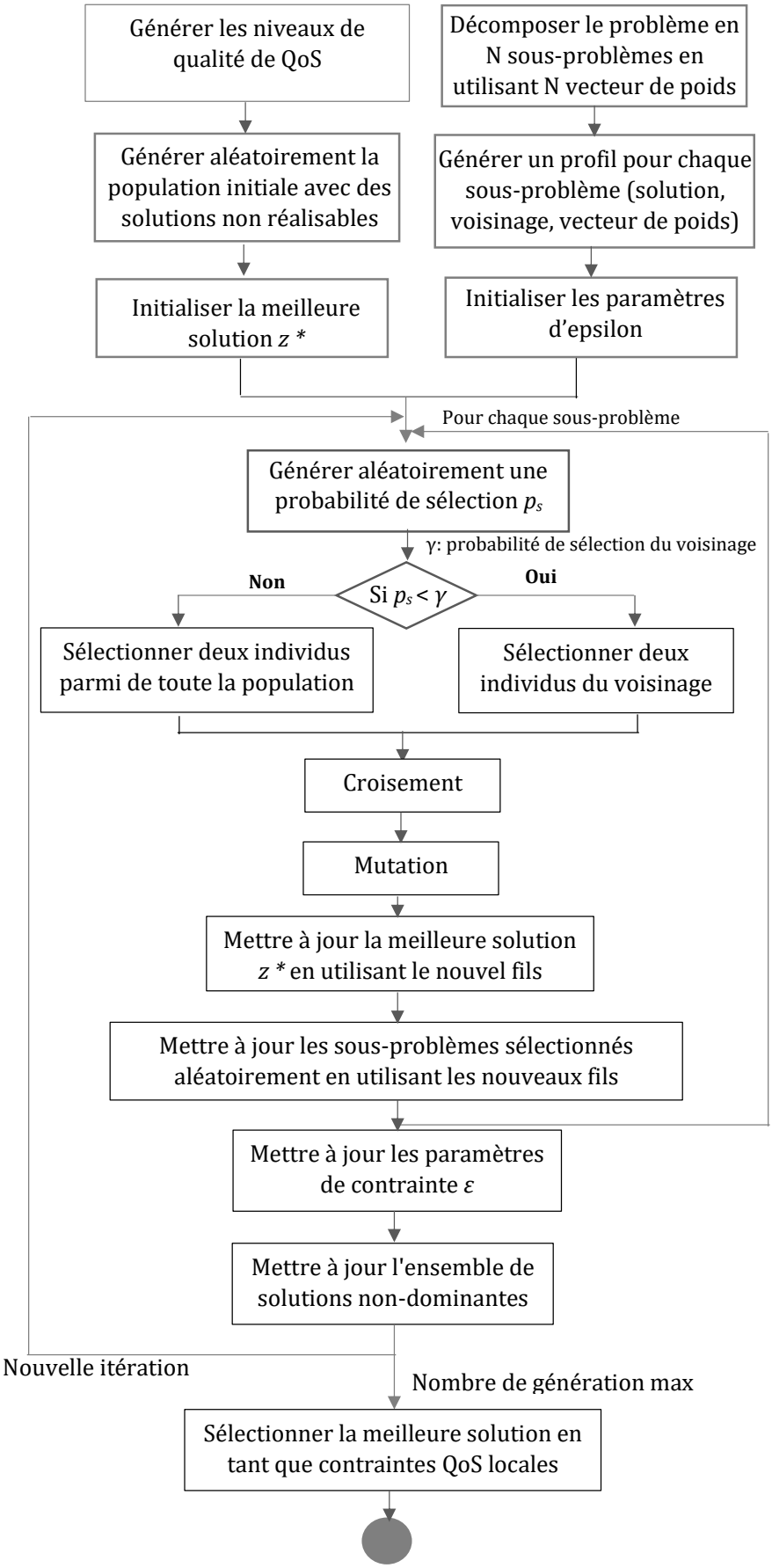
$$\begin{cases} max(\sum_{j=1}^{n}\sum_{k=1}^{r} p_{jk}^{z}) & \\ min(\sum_{j=1}^{n}\sum_{q_k}^{r} f_{jk}^{z}) & \text{Pour les attributs dynamiques} \\ Agg(L_{jk}^{z}) \leq E_k & \text{Pour les attributs négatifs} \\ Agg(L_{jk}^{z}) \geq E_k & \text{Pour les attributs positifs} \end{cases} \tag{4.9}$$

Les algorithmes évolutionnaires multi-objectifs sous contraintes (Constrained Multi-Objective Evolutionary Algorithm, CMOEA) représentent une solution efficace pour résoudre ce genre de problèmes dans un espace de recherche à grande échelle. Sur la base des études comparatives fournies dans [Chiandussi et al., 2012], [Fan et al., 2017] et [Fan et al., 2017], il est prouvé que la méthode MOEA/D-IEpsilon (MOEA/D-

Improved Epsilon) basée sur la décomposition en sous-problèmes constitue une solution efficace avec un bon équilibrage entre la diversité et la convergence. C'est pour cela, nous l'avons choisie pour résoudre le problème de décomposition des contraintes de QoS globales en contraintes locales. Pour simplifier, nous utilisons l'acronyme MOEA pour désigner cette méthode au lieu de MOEA/D-IEpsilon dans le reste de ce rapport.

La méthode proposée commence par générer la population initiale tout en considérant les solutions non-réalisables tolérées selon un seuil epsilon et en décomposant le problème de décomposition des contraintes de QoS globales en plusieurs sous-problèmes plus simples. Le niveau epsilon est ajusté dynamiquement à chaque génération en fonction du rapport entre les solutions faisables et totales (Ratio of Feasible Solutions, RFS) dans la population. Les solutions de ces sous-problèmes évoluent d'une population à l'autre en appliquant des opérateurs génétiques (sélection, croisement et mutation). Les nouvelles solutions obtenues remplacent les solutions les plus dégradées de la population actuelle à base d'epsilon. Ainsi, MOEA est composé de trois composants : **a)** la décomposition du problème en sous-problèmes, **b)** l'évolution de la population, et **c)** la gestion des contraintes.

Le diagramme suivant donne un aperçu de la méthode proposée.


Générer les niveaux de qualité de QoS
Décomposer le problème en N sous-problèmes en utilisant N vecteur de poids
Générer aléatoirement la population initiale avec des solutions non réalisables
Générer un profil pour chaque sous-problème (solution, voisinage, vecteur de poids)
Initialiser la meilleure solution z *
Initialiser les paramètres d'epsilon
Pour chaque sous-problème
Générer aléatoirement une probabilité de sélection p_s
γ: probabilité de sélection du voisinage
Non
Si $p_s < \gamma$
Oui
Sélectionner deux individus parmi de toute la population
Sélectionner deux individus du voisinage
Croisement
Mutation
Mettre à jour la meilleure solution z * en utilisant le nouvel fils
Mettre à jour les sous-problèmes sélectionnés aléatoirement en utilisant les nouveaux fils
Mettre à jour les paramètres de contrainte ε
Mettre à jour l'ensemble de solutions non-dominantes
Nouvelle itération
Nombre de génération max
Sélectionner la meilleure solution en tant que contraintes QoS locales

4.3.2.1 Décomposition du problème en sous-problèmes

La méthode MOEA commence par décomposer le problème global en N sous-problèmes en utilisant la méthode de Tchebycheff qui s'appuie sur N vecteurs de poids uniformément répartis $\left\{\lambda^1, \ldots, \lambda^N\right\}$, tels que : λ^m satisfait $\sum_{y=1}^{2} \lambda_y^m = 1$ et $\lambda_y^m \geq 0$ pour chaque $y \in \{1, 2\}$, 2 étant le nombre de fonctions objectives (utilité et fluctuation). Le $m^{\text{ème}}$ sous-problème est formulée ainsi en Équation 4.10 :

$$\begin{cases} \min g^{te}(\mathbf{x}|\lambda, z^*) = \max\{\lambda_1^m| - f_u(\mathbf{x}) - z_1^*|, \lambda_2^m|f_f(\mathbf{x}) - z_2^*|\} \\ \mathbf{x} \in Sol \end{cases} \tag{4.10}$$

Où : g^{te} est la fonction objective de Tchebycheff de la décomposition en sous-problèmes, Sol est l'ensemble de toutes les solutions possibles, f_u, f_f sont des fonctions objectives d'utilité et de fluctuation respectivement, $z^* = (z_1^*, z_2^*)$ est le point idéal, c'est-à-dire la meilleure solution obtenue jusqu'à présent, tels que : $z_1^* = min{-}f_u\{x|x \in Sol\}$ et $z_2^* = min f_f\{x|x \in Sol\}$.

Un sous-problème permet de faire évoluer une solution à partir de la population initiale. Il est défini par son profil qui contient : la meilleure solution obtenue x_m, la fonction objective du sous-problème définie par l'Équation 4.10, le vecteur de poids λ^m et l'ensemble de son voisinage B_m. Le profil d'un sous-problème est initialisé au début du processus d'optimisation à l'aide de l'algorithme 3. Dans ce dernier, et étant donné le nombre de sous-problèmes, 1) un vecteur de poids généré aléatoirement, et 2) un individu choisi au hasard de la population initiale est utilisé comme une solution initiale, et 3) l'ensemble de voisinage est choisi comme les sous-problèmes associés aux T vecteurs de poids les plus proches à son vecteur de poids.

4.3.2.2 Évolution de la population

La population de chaque sous-problème évolue d'une génération à l'autre dans le but d'améliorer les solutions en s'appuyant sur les opérateurs : le codage, la sélection, le croisement, la mutation et l'évaluation.

- **Codage d'individus :** le codage est une fonction qui permet de passer de la donnée réelle du problème de décomposition traité à la donnée utilisée par notre approche MOEA. Une solution $\mathbf{x}$ du problème, c'est-à-dire une instance chromosomique, consiste en une matrice qui décrit le niveau de qualité L_{jk}^z choisi pour chaque paramètre de QoS pour chaque cluster. Pour cela, un codage bi-dimensionnel est utilisé comme indiqué en Figure 4.4.

Algorithme 3 : Initialisation du profil des sous-problèmes

Entrées :
N : le nombre de sous-problèmes, population initiale,
T : taille du voisinage.
Sorties : N sous-problèmes définis via des triplets : (x_m : la solution initiale, λ^m : vecteur de poids, B_m : voisinage)

Début
pour *chaque sous-problème* m *(*$1 < m < N$*)* **faire**
  Générer au hasard un vecteur de poids $\lambda^m = (\lambda_1^m, \lambda_2^m)$
  Choisir au hasard une solution initiale x_m à partir de la population.
fin
pour *chaque sous-problème* m *(*$1 < m < N$*)* **faire**
  Calculer la distance euclidienne entre λ^m et les vecteurs de poids des $N-1$ autres sous-problèmes.
  Les sous-problèmes associés aux T vecteurs de poids les plus proches de λ^m sont choisis pour construire l'ensemble de voisinage B_m.
fin
Fin

	C_1	C_2	C_3		C_n
$\mathbf{q}^1$	L_{11}^a	L_{21}^b	L_{31}^c	$\cdots$	L_{n1}^e
$\mathbf{q}^2$	L_{12}^f	L_{22}^g	L_{32}^h	$\cdots$	L_{n2}^i
$\mathbf{q}^3$	L_{13}^j	L_{23}^o	L_{33}^t	$\cdots$	L_{n3}^u
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{q}^r$	L_{1r}^v	L_{2r}^w	L_{3r}^x	$\cdots$	L_{nr}^y

FIGURE 4.4 – Instance chromosomique

- **Évaluation des solutions :** les solutions sont évaluées via une fonction d'évaluation qui permet de déterminer leur pertinente. Dans notre approche, la fonction d'évaluation d'une solution x vise à calculer la valeur de la fonction objective composée de la valeur de l'utilité et de la fluctuation de x dans la population. Elle est définie par le vecteur : $(-f_u(x), f_f(x))$, tels que : $f_u(x)$ et $f_f(x)$ représentent l'utilité et la fluctuation de x respectivement. La comparaison entre deux solutions x_1 et x_2 selon cette fonction bi-objective est réalisée en utilisant la dominance. On dit que x_1 domine x_2,i.e., $x_1 \leq x_2$, si et seulement si :

  a- $f_u(x_1) \geq f_u(x_2)$ et $f_f(x_1) \leq f_f(x_2)$
  b- $f_u(x_1) > f_u(x_2)$ ou $f_f(x_1) < f_f(x_2)$

  C'est-à-dire que la solution x_1 est au moins aussi bonne que x_2 sur tous les

objectifs et, x_1 est strictement meilleure que x_2 sur au moins un objectif.

- **Opérateur de sélection :** l'opérateur de sélection est chargé de définir quels seront les individus de la population courante qui vont être réutilisées dans la population de la prochaine génération et vont servir de parents pour les opérateurs de croisement et de mutation. La stratégie de sélection par tournois est adoptée comme opérateur de sélection dans notre approche. Elle consiste à sélectionner les candidats les plus aptes de la génération actuelle pour les considérer dans la génération suivante. Pour cela, une partition d'individus de la population est choisie au hasard et des tournois sont organisés entre eux. Le vainqueur de chaque tournoi est celui avec la meilleure fonction d'évaluation. Il est sélectionné pour faire partie de la prochaine génération.
- **Opérateur de croisement :** le croisement est un opérateur qui permet la production des chromosomes qui héritent partiellement des caractéristiques de leurs parents. Il s'appuie sur la manipulation des structures de chromosomes parents pour en produire de nouveaux. Cet opérateur est utilisé afin d'enrichir la diversité de la population et permettre sa convergence. En raison de l'utilisation du codage bi-dimensionnel, la méthode basée sur l'échange de blocs est utilisée comme opérateur de croisement dans notre approche. Il consiste à échanger des blocs de taille aléatoire entre deux parents sélectionnés. Un exemple est illustré dans la Figure 4.5 avec un échange de blocs 2X3.

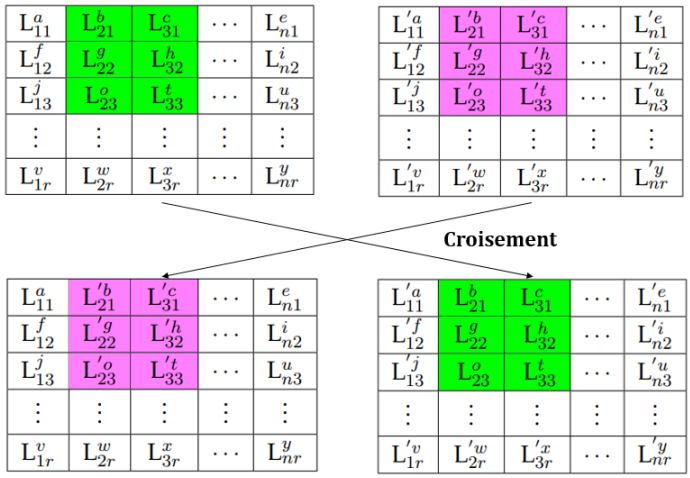


FIGURE 4.5 – Opérateur de croisement

- **Opérateur de mutation :** la mutation consiste simplement en l'inversion ou le remplacement d'un gène d'un chromosome par un autre d'une manière aléatoire. Elle apporte, alors, une faible modification aux individus pour permettre la diversité de la population. Pour notre problème, l'opérateur de mutation consiste

à choisir aléatoirement un niveau de qualité L_{jk}^{z} qui représente un gène dans le chromosome à muter et le remplacer par un autre niveau de qualité à partir de l'ensemble de niveaux de qualité QL_{jk} défini pour le même cluster C_j et le même paramètre de QoS q^k. Un exemple est donné en Figure 4.6.

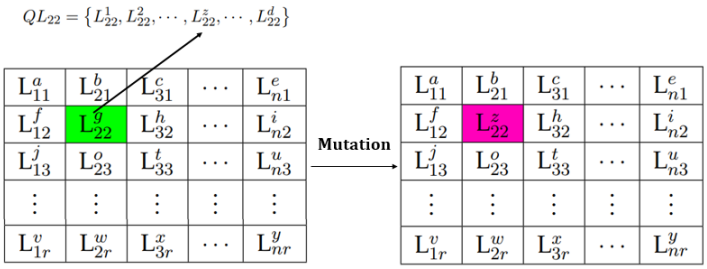


FIGURE 4.6 – Opérateur de mutation

4.3.2.3 Gestion des contraintes

Contrairement aux algorithmes génétiques traditionnels qui n'explorent que les régions des solutions réalisables et se contentent de la manipulation des solutions réalisables, notre algorithme considère les solutions non-réalisables. Cela permet de traverser les régions infaisables, ce qui permet d'aboutir à des solutions intéressantes via le croisement et la mutation des solutions non-réalisable, et par conséquent, garantir la diversification de l'exploration des solutions.

En plus, il permet d'éviter le blocage de processus de recherche dans un optimum local. Pour ce faire, il se base sur la relaxation des contraintes du problème. Cette relaxation est contrôlée en utilisant un niveau epsilon ε.

Donc, le troisième composant de notre approche est le mécanisme de gestion des contraintes de QoS globales. Il utilise une fonction de violation de contraintes ϕ décrite en Équation 4.11. Une solution x est faisable (réalisable) si et seulement si $\phi(x) = 0$, et non-réalisable sinon.

$$\phi(\mathbf{x}) = \sum_{i=1}^{p} |\min\left(Sup_i(\mathbf{x}) - E_k, 0\right)| + \sum_{j=1}^{q} \left|\max\left(Inf_j(\mathbf{x}) - E_k^{'}, 0\right)\right| \tag{4.11}$$

Tels que : Sup, Inf sont les contraintes de QoS globales supérieures et inférieures sous la forme $Sup_i(\mathbf{x}) \geq E_k$, $Inf_j(\mathbf{x}) \leq E_k^{'}$, $i \in [1, q]$, $j \in [1, p]$, p et q représentent le nombre de contraintes supérieures et inférieures respectivement.

Lors des premières itérations, un seuil de violation est toléré pour une valeur epsilon assez grande qui sera mise à jour (diminuée) selon le ratio RFS de la population de la

génération courante g comme indiqué dans l'Équation 4.12.

$$\phi(x) \leq \varepsilon(g) \tag{4.12}$$

Le paramétrage de la valeur de $\varepsilon(g)$ est donné via les règles de l'Équation 4.13. Ces règles permettent d'envisager les solutions irréalisables au cours de l'évolution de la population pour une meilleure convergence.

$$\varepsilon(g) = \begin{cases} R_1 : \phi\left(\mathbf{x}^{\theta}\right), & \text{Si } g = 0 \\ R_2 : (1-\tau) * \varepsilon(g-1), & \text{Si } r_g < \alpha \text{ et } g < T_c \\ R_3 : (1+\tau) * \phi_{\max}, & \text{Si } r_g \geq \alpha \text{ et } g < T_c \\ R4 : 0, & \text{Si } g \geq T_c \end{cases} \tag{4.13}$$

- $\phi_g\left(\mathbf{x}^{\theta}\right)$ est la violation globale des contraintes des θ tops individus dans la population initiale,
- r_g est le ratio RFS dans la $g^{ème}$ génération,
- $\tau \in [0, 1]$ est utilisé pour contrôler la vitesse de réduction de la relaxation des contraintes et contrôler le facteur d'échelle multiplié par la violation de contrainte globale maximale,
- $\alpha \in [0, 1]$ permet de contrôler la préférence de recherche entre les régions réalisables et irréalisables,
- $\phi_{\max}$ est la violation de contrainte de QoS globale maximale trouvée jusqu'à présent,
- $\varepsilon(g)$ est mis à jour jusqu'à ce que le compteur de générations g atteigne la génération de contrôle T_c lorsque $\varepsilon(g) = 0$

La meilleure solution obtenue x_m du $m^{ème}$ sous-problème est mise à jour chaque fois qu'un nouvel individu est obtenu. L'algorithme 4 illustre le processus de mise à jour qui prend en compte à la fois la fonction objective et la fonction de violation de contraintes de QoS globales. La solution actuelle x_m est remplacée par un nouvel individu y_m dans trois cas : **a)** les deux contraintes de violation des deux individus respectent la valeur du niveau epsilon et la valeur de la fonction objective de y_m est meilleure que celle de x_m, **b)** les deux ont la même valeur de contrainte de violation et y_m est plus optimal que x_m, et **c)** la fonction de violation des contraintes de y_m est plus faible que celle de

x_m.

Algorithme 4 : Mise à jour des solutions des sous-problèmes

Entrées :
x_m : la solution courante pour le $m^{ème}$ sous-problème,
y_m : l'individu nouvellement obtenu,
$\varepsilon(g)$: la valeur actuelle du epsilon.

Début
  si $(\phi(\mathbf{y}_m) \leq \varepsilon(g)\ ET\ \phi(\mathbf{x}_m) \leq \varepsilon(g))$ **alors**
    si $(g(\mathbf{y}_m|\lambda^m, z^*) \leq g(\mathbf{x}_m|\lambda^m, z^*))$ **alors**
      La solution $\mathbf{x}_m$ est remplacée par la solution $\mathbf{y}_m$
    fin
  fin
  si $(\phi(\mathbf{y}_m) == \phi(\mathbf{x}_m))$ **alors**
    si $(g(\mathbf{y}_m|\lambda^m, z^*) \leq g(\mathbf{x}_m|\lambda^m, z^*))$ **alors**
      La solution $\mathbf{x}_m$ est remplacée par la solution $\mathbf{y}_m$
    fin
  fin
  si $(\phi(\mathbf{y}_m) < \phi(\mathbf{x}_m))$ **alors**
    La solution $\mathbf{x}_m$ est remplacée par la solution $\mathbf{y}_m$
  fin
Fin

4.3.2.4 Algorithme global pour la décomposition des contraintes de QoS globales

L'approche globale pour la décomposition des contraintes de QoS globales est donnée par l'algorithme 5. Il se compose de trois étapes principales. Dans la première étape, les paramètres de l'algorithme sont initialisés, la population initiale est générée et le problème principal est décomposé. La deuxième étape consiste à faire évoluer les populations des sous-problèmes à l'aide des opérateurs génétiques (sélection, croisement, mutation).

La dernière étape considère les mises à jour des paramètres de l'algorithme : le niveau epsilon ainsi que les mises à jour des profils de sous-problèmes (les vecteurs λ, les solutions courantes et les voisinages).

Algorithme 5 : MOEA/D-IEpsilon : Décomposition des contraintes de QoS globales

Entrées : N : le nombre de sous-problèmes,
$T_{\max}$: le nombre maximum de générations,
N vecteurs de poids : $\lambda^1, \ldots, \lambda^N$,
γ : probabilité de sélection du voisinage,
n_r : le nombre maximal de sous-problèmes mis à jour au cours d'une itération,
τ, α, T_c, r_g et θ pour le paramétrage du epsilon.
Sorties : n contraintes de QoS locales

Début
Générer la population initiale $P = \{x_1, \ldots, x_N\}$
Décomposer le problème en N sous-problèmes
Initialiser $\varepsilon(0), r_g$, $\phi_{\max}$ et $z^* = (z_1^*, z_2^*)$
Mettre $iter = 0$ et $g = 0$.
tant que $iter \leq T_{\max}$ **faire**
  pour *(*$m \leftarrow 1$ *à* N*)* **faire**
    Générer la probabilité de sélection p_s dans [0,1]
    si *(*$p_s \leq \gamma$*)* **alors**
      Sélectionner deux parents $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$ du voisinage B_m
    fin
    sinon
      Sélectionner deux parents $\mathbf{x}_{p1}$, $\mathbf{x}_{p2}$ de toute la population
    fin
    Produire un nouveau fils via l'opérateur de croisement $\mathbf{y}^{cr}$ avec une probabilité p_c.
    Appliquer l'opérateur de mutation sur $\mathbf{y}^{cr}$ avec une probabilité p_{mu} et produire $\mathbf{y}^{mu}$
    $iter = iter + 1$; Mettre à jour $\phi_{\max}$
    Mettre à jour le point idéal z^* : **si** *(*$z_1^* < f_u(y_{mu})$*)* **alors**
      $z_1^* = f_u(y_{mu})$
    fin
    si *(*$z_2^* > f_f(y_{mu})$*)* **alors**
      $z_2^* = f_f(y_{mu})$
    fin
    Mettre à jour n_r sous-problèmes sélectionnés au hasard en utilisant le nouvel fils $\mathbf{y}^{mu}$ avec l'algorithme 4
  fin
  $g = g + 1$; Mettre à jour r_g ; Mettre à jour $\varepsilon(g)$
  $NS =$ les solutions non dominées dans (NS $\cup P$)
fin
Sélectionnez la meilleure solution de NS comme contraintes locales de QoS
Fin

4.3.2.5 Application : Calcul des contraintes de QoS locales via MOEA dans le scénario de dépassement

Pour trouver les contraintes de QoS locales à utiliser, nous commençons tout d'abord par transformer les structures complexes du processus en structures séquentielles simples comme donné dans [Wang et al., 2011]. Cela permet de décomposer les contraintes globales de QoS en contraintes locales d'une manière récursive.

Le processus de dépassement global est donc transformé en processus séquentiel, et cela, en créant des tâches composées pour assembler les structures complexes. Par exemple, $T_{768} = ((T_7$ [séquentiel] $T_8)$ [parallèle] $T_6)$. Les étapes de transformation sont illustrées dans la Figure 4.7.

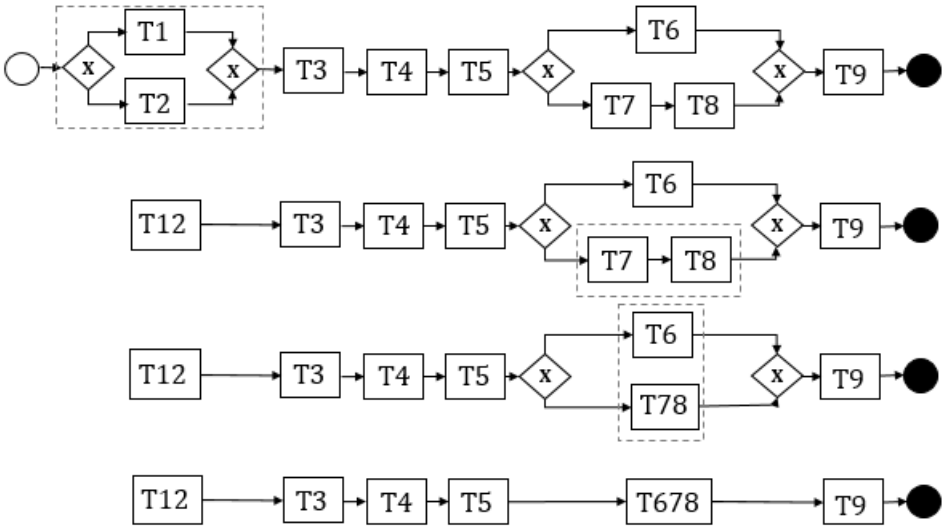


FIGURE 4.7 – Transformation en processus séquentiel

Après avoir calculé les niveaux de qualité pour les différences tâches élémentaires composant le processus (Tableau 4.4), les niveaux de qualité des tâches composées sont obtenus en utilisant l'agrégation (cf. Tableau 4.3) des niveaux de qualité des tâches élémentaires qu'ils les composent, et cela, selon la structure et le lien qui les relie et le type du paramètre de QoS. Par exemple, pour la tâche composée T_{12}=(T_1 [parallèle] T_2), les niveaux de qualité pour le temps de réponse sont obtenus en utilisant l'agrégation *maximum* entre les niveaux de qualité des tâches T_1 et T_2, i.e., pour le premier niveau de qualité $L^1_{12,1} = max(\{L^1_{1,1}, L^1_{2,1}\} = max(\{51.33, 39.5\} = 51.33$ et de même pour les autres niveaux de qualité. En ce qui concerne les niveaux de qualité pour le débit, l'agrégation *minimum* est utilisée. Par exemple, le premier niveau de qualité $L^1_{12,2} = min(\{L^1_{1,2}, L^1_{2,2}\} = min(\{10.66, 12\} = 10.66$ et de même pour les autres niveaux de

qualité.

Pour l'évaluation des niveaux de qualité des tâches composées, puisque la fonction objectif vise à maximiser l'utilité des niveaux de QoS et à minimiser leur fluctuation, nous fixons l'utilité agrégée de chaque niveau de qualité d'une tâche composée à la valeur d'utilité minimale des tâches qui la composent et sa fluctuation à la valeur de fluctuation maximale. Nous pouvons généraliser ce calcul comme suit : Soit T_c une tâche composite et $\{T_1, T_2, \ldots, T_y\}$ l'ensemble de ses tâches élémentaires, l'utilité et la fluctuation de $z^{ème}$ niveau de qualité de $k^{ème}$ attribut de QoS est donné selon l'Équation 4.14.

$$\begin{cases} p_k^z(T_c) = min(p_{jk}^z), 1 \leq j \leq y \text{ (utilité)} \\ f_k^z(T_c) = max(f_{jk}^z), 1 \leq j \leq y \text{ (fluctuation)} \end{cases} \tag{4.14}$$

Les niveaux de qualité des taches composées et leurs évaluations sont affichés dans les Tableaux 4.5 et 4.6.

Tâche	L_{j1}^1	p_{j1}^1	f_{j1}^1	L_{j1}^2	p_{j1}^2	f_{j1}^2	L_{j1}^3	p_{j1}^3	f_{j1}^3
T_{12}	51.33	0.16	0.23	55.16	0.33	0.5	64.66	1	1
T_{78}	98.82	0.33	0.28	102.66	0.5	0.42	132.33	1	1
T_{786}	98.82	0.16	0.28	102.66	0.5	0.65	132.33	0.83	1

TABLE 4.5 – Niveaux de qualité et leurs évaluations pour les tâches composées - Temps de réponse

Tâche	L_{j2}^1	p_{j2}^1	f_{j2}^1	L_{j2}^2	p_{j2}^2	f_{j2}^2	L_{j2}^3	p_{j2}^3	f_{j2}^3
T_{12}	10.66	0.83	1	12.83	0.66	0.63	18.83	0.13	0.11
T_{78}	10	0.83	1	12	0.5	0.66	16.66	0.16	0.28
T_{786}	10	0.83	1	12	0.5	0.66	16.66	0.12	0.28

TABLE 4.6 – Niveaux de qualité et leurs évaluations pour les tâches composées - Débit

Après la transformation de processus, plusieurs décompositions de contraintes seront opérées pour déterminer les contraintes locales pour chaque tâche simple, et cela, en déterminant les contraintes locales des tâches composées qui vont servir de contraintes globales pour les tâches simples qui les composent.

Premièrement, nous appliquons l'algorithme MOEA sur le processus $P_1 = \{T_{12}, T_3, T_4, T_5, T_{786}, T_9\}$ sous les contraintes de QoS du processus global de dépassement pour obtenir les contraintes locales pour chaque tâche dans ce processus comme indiqué dans le Tableau 4.7.

Tâche	e_{j1}	e_{j2}
T_{12}	64.66	12.83
T_3	58.5	16.5
T_4	45.33	16.16
T_5	43.66	16
T_{786}	102.66	12
T_9	40.66	18

TABLE 4.7 – Contraintes de QoS locales pour P_1

Ensuite, nous appliquons MOEA sur $P_2 = \{T_1, T_2\}$ sous les contraintes : 64.66 ms pour le temps de réponse et 12.83 r/s pour le débit. Le résultat est donné en Tableau 4.8.

Tâche	e_{j1}	e_{j2}
T_1	61.83	18.83
T_2	45.33	12.83

TABLE 4.8 – Contraintes de QoS locales pour P_2

Après cela, le résultat de l'application de MOEA sur $P_3 = \{T_6, T_{78}\}$ sous les contraintes : 102.66 ms et 12 r/s est donné en Tableau 4.9.

Tâche	e_{j1}	e_{j2}
T_6	50	14.83
T_{78}	102.66	12

TABLE 4.9 – Contraintes de QoS locales pour P_3

La dernière étape consiste à appliquer MOEA sur le processus $P_4 = \{T_7, T_8\}$ sous les contraintes : 102.66 ms et à 12 r/s. Le résultat est donné en Tableau 4.10.

Tâche	L_{j1}^{opt}	L_{j2}^{opt}
T_7	46.66	16.16
T_8	53.16	16.66

TABLE 4.10 – Contraintes de QoS locales pour P_4

4.3.3 Sélection locale

Une fois que les contraintes de QoS locales pour chaque attribut de QoS q^k sont obtenues, l'avatar initiateur transmet ces valeurs aux avatars élus correspondants. Les contraintes de QoS locales reçues par les élus leur servent de bornes pour sélectionner l'avatar ayant l'utilité la plus élevée et la fluctuation la plus faible de leurs clusters d'une manière parallèle et indépendante. Formellement, soit $E = \{e_{j1}, e_{j2}, \ldots, e_{jr}\}$ l'ensemble des contraintes de QoS locales, l'avatar sélectionné A_i de chaque cluster C_j doit satisfaire les conditions suivantes :

$$\begin{cases} \max(U(A_{ij}) - F(A_{ij})) & \\ q_{ij}^k \leq e_{jk} & \text{Pour les attributs négatifs} \\ q_{ij}^k \geq e_{jk} & \text{Pour les attributs positifs} \end{cases}$$

Après avoir sélectionné les avatars les plus aptes, chaque élu envoie son choix à l'initiateur pour construire la solution de composition finale.

L'algorithme 6 donne la méthode de sélection locale effectuée par les avatars élus dans chaque cluster C_j en parallèle.

4.3.3.1 Application : Sélection locale dans le scénario de dépassement

En appliquant une sélection locale selon les contraintes locales obtenues précédemment, les avatars qui respectent ces contraintes avec la meilleure fonction d'évaluation sont sélectionnées de chaque cluster respectivement :

A_8 pour T_1, A_{13} pour T_2, A_{12} pour T_3, A_7 pour T_4, A_1 pour T_5, A_{15} pour T_6, A_{16} pour T_7, A_3 pour T_8, et A_1 pour T_9.

Algorithme 6 : Méthode de sélection locale

Entrées : r contraintes de QoS locales
Sorties : Avatar sélectionné

Début
pour *(chaque avatar A_i dans le cluster C_j)* **faire**
  $U(A_{ij}) \leftarrow CalculerUtilité(A_{ij})$
  $F(A_{ij}) \leftarrow CalculerFluctuation(A_{ij})$
  $Calculer(U(A_{ij}) - F(A_{ij}))$
fin
$TrierAvatars(A_{ij})$
pour *(chaque A_i dans C_j)* **faire**
  pour *(chaque q^k)* **faire**
    si *(q^k est positif ET non($q_{ij}^k \geq e_{jk}$))* **alors**
      Break
    fin
    si *(q^k est négatif ET non($q_{ij}^k \leq e_{jk}$))* **alors**
      Break
    fin
  fin
  $Sélectionner(A_{ij})$, Break
fin
Fin

4.4 Contribution IV : Prédiction des contraintes de QoS locales basée sur SVM

L'approche proposée précédemment permet de déterminer les contraintes locales en se basant sur la collaboration entre l'avatar initiateur et les élus. Pour optimiser les communications entre ces avatars et ainsi le temps que cela implique, nous proposons d'étendre l'approche présentée précédemment en s'orientant vers les méthodes d'apprentissage automatique (machine learning) pour la prédiction des contraintes de QoS locales d'un processus donné à partir des données de QoS historiques et des exécutions antérieures. Cette nouvelle approche est basée sur un modèle de raisonnement par cas (Case Based Reasoning, CBR) et la méthode de régression par machine à vecteurs de support (Support Vector machine, SVM) en considérant le contexte d'exécution (localisation et plage horaire) du processus pour lequel la prédiction des contraintes de QoS locales est effectuée. Le problème à traiter peut être formulé comme suit : étant donné un processus P composé de n tâches abstraites $P = \{T_1, T_2, \ldots, T_n\}$ et soumis à r exigences de QoS globales $E = \{E_1, E_2, \ldots, E_r\}$, le but est de prédire les contraintes

de QoS à utiliser pour la sélection locale.

L'approche proposée comprend principalement trois phases comme donné en Figure 4.8 : **1)** la génération des cas de base permet de récupérer les cas les plus similaires au cas à exécuter à partir des matrices des cas historiques sauvegardés au niveau de la base de connaissance de l'avatar initiateur en utilisant un modèle de raisonnement par cas. Cette phase est basée sur l'utilisation d'une distance de similarité pondérée en termes de localisation et d'horaire pour construire l'ensemble des données de QoS d'entraînement et d'apprentissage, **2)** le prétraitement des données pour l'entraînement et l'apprentissage, quant à elle, permet de regrouper les données de QoS historiques générées dans la première phase et les répartir dans le temps selon une fenêtre temporelle glissante.

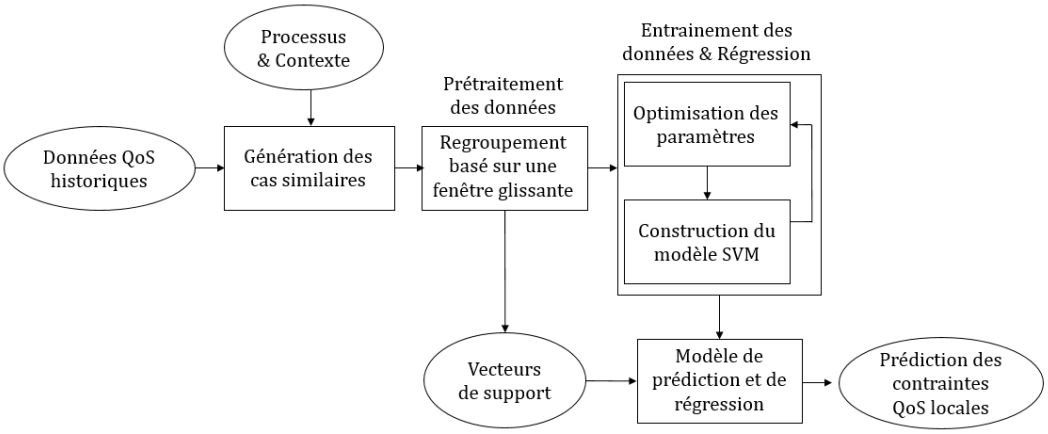


FIGURE 4.8 – Aperçu de l'approche de prédiction par régression SVM

Elle permet de préparer les données récupérées et les prétraiter pour les utiliser dans la phase d'apprentissage et de prédiction des contraintes de QoS locales, et **3)** l'apprentissage et de prédiction des contraintes de QoS locales à partir de l'entraînement des données des cas similaires prétraités en utilisant la méthode de régression SVM.

4.4.1 Génération de cas de base similaires via CBR

Le raisonnement basé sur les cas (CBR) [Watson, 1999] [Kolodner, 2014] est une méthodologie de raisonnement qui se base sur l'utilisation des résultats et des connaissances acquises des expériences antérieures dont le contexte est semblable à celui de la situation actuelle pour construire une nouvelle solution qui, à son tour, sera conservée dans la base de connaissance et s'ajoutera à la liste des expériences. Les connaissances et les expériences précédentes sont formalisées sous forme de situations appelées cas. Cette méthode comprend quatre étapes [Aamodt and Plaza, 1994] comme indiqué dans

la Figure 4.9.

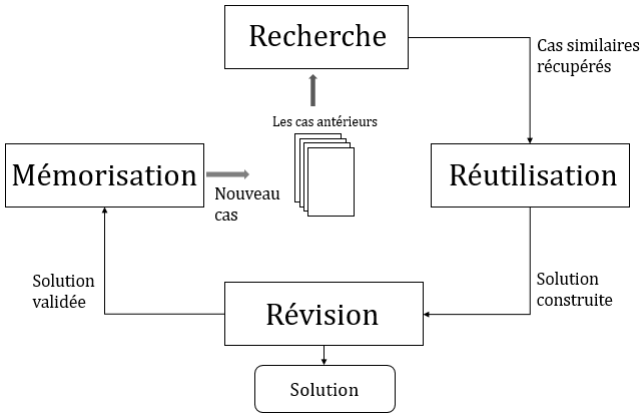


FIGURE 4.9 – Le processus de raisonnement basé sur les cas

1. La recherche : consiste à rechercher et à sélectionner les cas les plus similaires au cas du problème cible à résoudre. Ces cas seront utilisés comme base pour la résolution du nouveau problème cible [Stahl, 2003].
2. La réutilisation : elle permet d'extraire et d'utiliser les informations contenues dans les différents cas sélectionnés dans l'étape de recherche pour construire une solution adaptée au problème à résoudre.
3. La révision : elle consiste à réviser la nouvelle solution construite et à tester son exactitude.
4. La mémorisation : si la solution testée dans l'étape précédente est jugée exacte alors elle est ajoutée à la base de connaissance du système pour enrichir ses expériences.

Représentation des cas

Dans la littérature, il n'existe pas une définition commune du terme *cas*, cependant, au sens large, un cas peut être défini comme la composante élémentaire des systèmes de raisonnement à base de cas. En effet, la définition d'un cas est très relative au format de représentation utilisée pour le modéliser. Plusieurs formalismes peuvent être utilisés pour la représentation et la modélisation des cas [Bergmann, 2003][Lejri and Tagina, 2012], cependant, le formalisme le plus couramment utilisé est la représentation <Problème, Solution> et c'est la représentation que nous avons également adopté dans ce travail. Par conséquent, la description d'un cas de base doit contenir une description du problème : le processus à exécuter décrit par sa référence RDF dans la base de connaissance de l'avatar, et sa solution : contraintes de

QoS locales pour chaque tâche composant ce processus et cela pour chaque attribut de QoS sous forme d'une matrice dont les lignes correspondent aux tâches et les colonnes aux attributs de QoS. Chaque cas est également indexé par deux facteurs contextuels : localisation et plage horaire.

Le format de modélisation d'un cas Cas_i est défini comme suivant :

$$Cas_i =< PR_i, LQoSCM_i||H_i, L_i >^{TT_i}$$

Tels que : PR (Process RDF) indique le descriptif RDF du processus à exécuter, $LQoSCMi$ (Local QoS Constraints Matrix) est la matrice solution, H (Heure) indique l'heure à laquelle le cas est exécuté et L (Localisation) indique les coordonnées géographiques où se trouve l'objet de l'avatar au moment de l'exécution du cas pour déterminer le contexte dans lequel le processus est exécuté. Concernant TT (Target Time), il représente la date d'exécution des cas afin de définir les regroupements des données d'entraînement et d'apprentissage pour le modèle de prédiction selon leur ordre chronologique.

Récupération des cas similaires

Cette phase consiste à rechercher et à calculer les h cas historiques les plus similaires au cas cible du problème à résoudre. Dans les systèmes à large échelle, nous recommandons de limiter le nombre de cas sauvegardés dans la base de connaissances à 1500 cas au maximum, ce qui permet d'avoir une information suffisante sur le processus et limiter les calculs. Pour h, nous avons choisi dans ce travail de le fixer à 2/3 du nombre de cas sauvegardés. Nous avons trouvé à travers les expérimentations que ce nombre est suffisant pour prédire des contraintes de QoS locales qui mènent à des compositions avec une bonne optimalité.

Avant d'entamer la phase de recherche, il est important de définir le format de représentation du cas cible :

$$Cas_0 =< PR_0, ?LQoSCM_0||H_0, L_0 >^{TT_0}$$

Tel que : $?LQoSCM_0$ est la matrice de contraintes de QoS locales à prédire.

La récupération des cas similaires de la base de connaissance de l'avatar est décrite principalement en deux étapes : **1)** la récupération des cas correspondant au processus à exécuter, i.e., les cas historiques du même processus décrit par le RDF PR_0, et **2)** le calcul de la similarité entre les cas récupérés et le cas cible pour sélectionner les h cas les plus similaires comme cas de base pour la prédiction des contraintes de QoS locales.

Pour le calcul de la similarité entre le cas cible Cas_0 et un cas Cas_i, nous utilisons une somme pondérée de deux distances comme donné en Équation 4.15 :

$$SIM(Cas_0, Cas_i) = w_1.dist_1 + w_2.dist_2 \tag{4.15}$$

Les coefficients w_1 et w_2 indique la préférence et le poids donné pour chacune des distances $dist_1$ et $dist_2$ respectivement, tel que : $w_1 + w_2 = 1$.

La distance $dist_1$ est basée sur la distance à vol d'oiseau (Équation 3.2)normalisée et calculée à partir des données géographiques contenues dans Cas_0 et Cas_i comme donné en Équation 4.16.

$$dist_1(Cas_0, Cas_i) = \frac{1}{D(Cas_0, Cas_i) + 1} \tag{4.16}$$

La distance $dist_2$ permet de calculer la distance normalisée en terme de plage horaire où le processus s'exécute. Elle est calculée en considérant les heures d'exécution de Cas_0 et Cas_i et elle est donnée en Équation 4.17.

$$dist_2(Cas_0, C_i) = \frac{\mid H_i - H_0 \mid}{24} \tag{4.17}$$

Prétraitement des cas de base

Après avoir récupéré les h cas les plus similaires, nous procédons au prétraitement de ces cas, considérés comme des échantillons d'enregistrements, pour la phase d'entraînement et d'apprentissage afin de pouvoir dériver un modèle de prédiction des contraintes de QoS locales via la méthode de régression SVM. Pour cela, nous divisons ces échantillons en plusieurs sous-séquences chronologiques avec une fenêtre glissante de taille fixe. Par conséquent, pour chaque tâche abstraite et chaque paramètre de QoS, un ensemble de vecteurs de support sont générés. En prenant t comme la taille de la fenêtre glissante, $h-t$ groupes de relations de cartographie peuvent être générés. Chaque fenêtre correspond à un point de données d'entraînement (exemple d'entraînement), tel que : les $t-1$ premières composantes sont utilisées comme base pour la prédiction de la $t^{ème}$ composante. Le modèle d'entraînement cherche à trouver une relation générique entre les $t-1$ premières composantes et la $t^{ème}$ composante. Les $h-t$ vecteurs de support générés pour l'entraînement sont modélisés comme suivant :

$$\{(X_z, Y_z)\}_{z=1}^{h-t}, \text{ tel que : } X_z \in R^{t-1} \text{ et } Y_z \in R.$$

Cette méthode de regroupement des cas selon une fenêtre glissante est illustrée en Figure 4.10. Cette dernière montre trois points de données d'apprentissage ($X_1 = \left\{e_{jk}^1, e_{jk}^2, ..., e_{jk}^{t-1}\right\}, Y_1 = e_{jk}^t$),($X_2 = \left\{e_{jk}^2, e_{jk}^3, ..., e_{jk}^t\right\}, Y_2 = e_{jk}^{t+1}$) et ($X_{h-t} = \left\{e_{jk}^{h-t}, e_{jk}^{h-t+1}, ..., e_{jk}^{h-1}\right\}, Y_{h-t} = e_{jk}^k$), tel que : e_{jk}^z est la contrainte de QoS locale de la $j^{ème}$ tâche et le $k^{ème}$ attribut de QoS du $z^{ème}$ cas.

4.4 Contribution IV : Prédiction des contraintes de QoS locales basée sur SVM

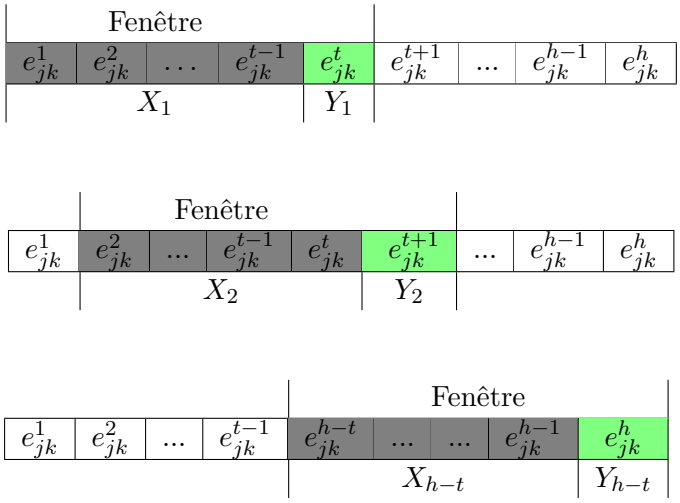


FIGURE 4.10 – Construction des vecteurs de support (X,Y)

La taille de la fenêtre est un paramètre très important, parce qu'une fenêtre d'une taille trop grande peut introduire du bruit avec une redondance d'informations et une taille trop petite peut entraîner une mauvaise qualité de prédiction.

Application : Récupération des cas similaires pour le processus de dépassement

Nous supposons que la base de connaissance de l'avatar du véhicule hôte contient 30 cas et nous modélisons le cas de dépassement pour lequel nous recherchons la matrice de contraintes locales comme suivant :
$C_0 =< AvatarOnt : Process_1, ?Matrice||14h, (43.34583, 1.17507) >^{21/08/2020}$

Ensuite, nous calculons la distance entre les cas historiques et le cas de dépassement C_0. Le résultat est donné dans le Tableau A.4 en Annexe A.

Une fois que les distances sont calculées, nous sélectionnons les 20 cas (qui représentent 2/3 du nombre de cas dans la base) qui possèdent les plus grandes similarités avec le cas cible comme cas de base pour l'entraînement de notre modèle de prédiction : C_{24}, C_7, C_{10}, C_{19}, C_7, C_6, C_{16}, C_{26}, C_{27}, C_{30}, C_{17}, C_{14}, C_{22}, C_{21},C_{18}, C_{12}, C_{11}, C_3, C_{28}, C_5.

Après la récupération des cas de base, nous extrairons les contraintes de QoS locales par tâche et par paramètre de QoS et nous les ordonnons par ordre chronologique comme illustré dans le Tableau A.5 en Annexe A.

À partir des cas historiques similaires au cas cible, nous générons les vecteurs d'apprentissage (X, Y) en utilisant une fenêtre glissante de taille égale à 4 (valeur inférieure à 30% des 20 cas similaires) pour les utiliser dans le modèle d'entraînement et d'apprentissage. Par exemple, pour le premier vecteur pour la tâche T_1 pour le temps de réponse,

nous avons les cas de base ordonnés d'une manière chronologique : 62.68, 61.15, 62.29, 58.75, 60.97, 59.56, 57.47, 59.02, 60.45, 57.78, 57.28, 57.78, 61.79, 58.67, 61.88, 60.66, 61.24, 62.53, 59.75, et 62.62. En appliquant une fenêtre glissante de taille 4, et par conséquent, nous obtenons l'observation : $X_1 = (62.68, 61.15, 62.29, 58.75), Y_1 = 60.97$.

4.4.2 Régression SVM

La régression SVM [Boser et al., 1992], appelée également séparateur à vaste marge, est une technique populaire d'apprentissage supervisé. Elle est, initialement, destinée à la résolution des problèmes de discrimination (classification), et puis, étendue pour inclure les problèmes de régression.

Le point fort de cette méthode est son fondement théorique solide et sa capacité à considérer de grandes quantités de données de plusieurs dimensions [Cristianini et al., 2000]. Elle présente de très bons résultats en pratique et ses performances sont généralement de même ordre, ou même supérieure, à celles d'un réseau de neurones [Sánchez A, 2003]. Elle est basée sur la théorie de l'apprentissage statistique respectant le principe de la minimisation des risques structurels (structural risk minimization, SRM) qui consiste à trouver une fonction de prédiction qui minimise la somme des erreurs d'apprentissage. Elle s'appuie sur l'utilisation d'un nombre limité d'exemples d'entraînement, i.e., un nombre d'entrées avec leurs sorties correspondantes, pour apprendre une fonction générique qui décrit le lien existant entre les entrées utilisées et leurs sorties. Par conséquent, la fonction apprise sera utilisée pour la prédiction de la sortie du système à l'instant présent.

Principe de la régression SVM

Le principe de l'apprentissage par régression SVM consiste à utiliser un échantillon de données $D = \{(X_z, Y_z)\}_{z=1}^{h-t}$, pour trouver une fonction $f : \mathbb{R}^{t-1} \to \mathbb{R}$ qui minimise l'erreur entre Y_z et $f(X_z)$ pour chaque point d'apprentissage. Dans le jeu d'échantillons d'apprentissage : $X_z \in \mathbb{R}^{t-1}$ est la valeur de la variable d'entrée et ses composantes sont appelées supports, et $Y_z \in \mathbb{R}$ est la valeur de sortie. La réalisation de cette méthode revient à la résolution d'un problème d'optimisation pour trouver les paramètres les plus adéquats pour construire la fonction de prédiction f. Pour cela, les données d'entrée sont, tout d'abord, mappées de leur espace original dans un espace de redescription à haute dimension appelé *espace de Hilbert* en utilisant une fonction non-linéaire $\emptyset(X)$. Par conséquent, le problème de prédiction est transformé en un problème de régression linéaire dans un espace d'une dimension supérieure à la dimension des X, et où il existe une solution linéaire calculée ainsi :

$$f(X) = \omega^T.\emptyset(X) + b \tag{4.18}$$

Tels que : ω^T est un vecteur transposé et b est un scalaire.

À travers ce problème, nous cherchons à minimiser l'erreur quadratique (ε) entre les Y_z et les $f(X_z)$ pour chaque point d'entraînement en utilisant une fonction de perte insensible (ε) (ε-insensitive loss function) qui est définie en 4.19.

$$L_\varepsilon(X,Y,f) = \begin{cases} 0 & if \mid Y - f(X) \mid \leq \varepsilon \\ \mid Y - f(X) \mid -\varepsilon & Sinon \end{cases} \tag{4.19}$$

Selon la théorie de l'apprentissage statistique, pour obtenir le vecteur ω^T et le terme b, la somme de la fonction de perte $L_\varepsilon(X,Y,f)$ doit être minimisée comme indiqué dans l'Équation 4.20.

$$min(\frac{1}{2}||\omega^2|| + C\sum_{z=1}^{h-t} L_\varepsilon(X_z, Y_z, f)) \tag{4.20}$$

Tel que : C est une constante d'une valeur positive qui mesure la pénalité imposée aux observations qui se trouvent en dehors de la marge epsilon (ε).

Cependant, il est difficile de trouver une fonction $f(X)$ dont la fonction de perte $L_\varepsilon(X,Y,f)$ existe pour tous les points d'apprentissage. Pour faire face à cela, les variables Slack ξ_z et ξ_z^* [1] sont introduites. Cela permet aux erreurs de régression d'exister avec une certaine marge jusqu'à la valeur de ξ_z et ξ_z^*, tout en satisfaisant les conditions requises. L'introduction des variables Slack transforme la formule de la fonction objective en une formule primale comme illustré dans l'Équation 4.21.

$$min[\frac{1}{2}||\omega^2|| + C\sum_{z=1}^{h-t}(\xi_z + \xi_z^*)] \text{ , tels que : } \begin{cases} (\omega^T.\emptyset(X_z) + b) - Y_z \leq \varepsilon + \xi_z & \forall z \\ Y_z - (\omega^T.\emptyset(X_z) + b) \leq \varepsilon + \xi_z^* & \forall z \\ \xi_z, \xi_z^* \geq 0 & \forall z \end{cases} \tag{4.21}$$

Le problème primal tel qu'il est décrit n'est pas facile à résoudre à cause de la grande dimension de l'espace de description. Par conséquent, il est converti en un problème dual en utilisant les multiplicateurs de Lagrange α et α^* [2] comme indiqué dans l'Équation 4.22.

$$max[-\tfrac{1}{2}\textstyle\sum_{z=1}^{h-t}\sum_{u=1}^{h-t}(\alpha_z^* - \alpha_z)(\alpha_u^* - \alpha_z)K(x_u, x) + \sum_{z=1}^{m}(\alpha_z^* - \alpha_z)y_z - \varepsilon\sum_{z=1}^{h-t}(\alpha_z^* + \alpha_z)]$$

$$\text{Tels que : } \begin{cases} \sum_{i=z}^{h-t}(\alpha_z - \alpha_z^*) = 0 \\ \alpha_z - \alpha_z^* \in [0, C] \end{cases} \tag{4.22}$$

$K(x_u, x)$ est appelée fonction Kernel [3]. Il existe essentiellement trois fonctions Kernel : linéaire, polynomiale et RBF (Radial Basis Function). Le Tableau 4.11 décrit les

1. http://apmonitor.com/wiki/index.php/Main/SlackVariables
2. http://www.mathoman.com/index.php/1650-multiplicateurs-de-lagrange
3. http://www.kernelmachines.org/

formules de calcul des différentes fonctions Kernel considérées.

Fonction Kernel	Formule
Linéaire	$K(X_u, X) = X_u^t.X$
Polynomiale	$K(X_u, X) = (1 + X_u^t.X)^p$, tel que : p est le degré de polynôme
RBF	$K(X_u, x) = exp(\frac{-\lvert X_u - X\rvert^2}{\delta^2})$

TABLE 4.11 – Fonctions Kernel

En résolvant le problème dual de programmation quadratique de l'Équation 4.22, les vecteurs α et α^* sont obtenus. Par conséquent, la nouvelle fonction de régression est donnée dans l'Équation 4.23.

$$y = f(X) = \sum_{z=1}^{h-t}(\alpha_z^* - \alpha_z)K(X_z, X) + b$$

$$\text{Tels que : } \begin{cases} b = Y_z - \sum_{i=z}^{h-t}(\alpha_z^* - \alpha_z)K(X_z, X) - \varepsilon & \alpha_z \in [0, C] \\ b = Y_z - \sum_{z=1}^{h-t}(\alpha_z^* - \alpha_z)K(X_z, X) + \varepsilon & \alpha_z^* \in [0, C] \end{cases} \tag{4.23}$$

Le principe générique de la régression SVM est illustré en Figure 4.11 [Yan and Zhi-Zhong, 2017].

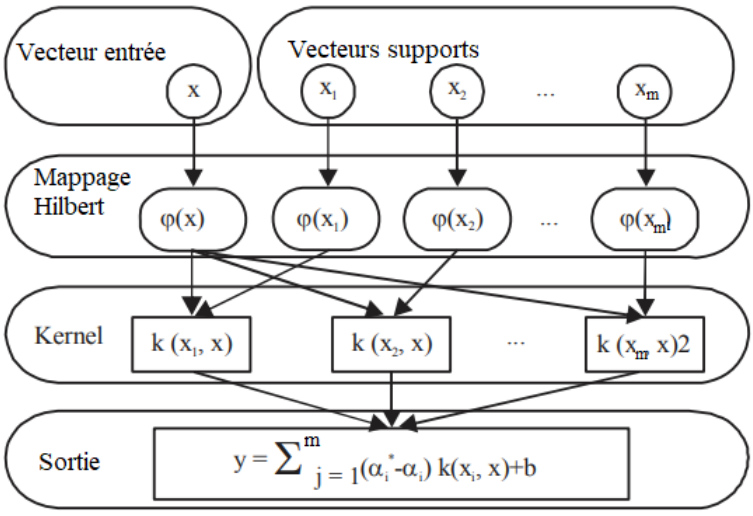


FIGURE 4.11 – Processus de régression SVM

L'algorithme 7 donne un aperçu global de la méthode de prédiction des contraintes de QoS locales de la récupération des cas de base les plus similaires au cas à résoudre,

à la construction des points de données d'entraînement pour l'élaboration du modèle de régression SVM utilisé pour la prédiction.

Application : Prédiction des contraintes de QoS locales pour le scénario de dépassement

Les Tableaux 4.12 et 4.13 illustrent les formules des modèles de prédiction obtenus après l'apprentissage pour chaque tâche et pour chaque paramètre de QoS (temps de réponse et débit respectivement). Par exemple, pour la première tâche, et afin de prédire la contrainte locale de temps de réponse, nous avons obtenu la fonction $f(X) = 0.003.x_1 - 0.0003.x_2 + 0.005.x_3 + 0.956.x_4 + 2.08$, tels que : $x_1, x_2, x_3 et x_4$ représentent les valeurs de QoS de temps de réponse des quatre derniers cas similaires les proches chronologiquement.

Tâche	Modèle temps de réponse
T_1	$0.003.x_1 - 0.0003.x_2 + 0.005.x_3 + 0.956.x_4 + 2.08$
T_2	$0.008.x_1 - 0.002.x_2 - 0.004.x_3 + 0.954.x_4 + 2$
T_3	$-0.017.x_1 + 0.023.x_2 - 0.022.x_3 + 0.926.x_4 + 5.183$
T_4	$-0.012.x_1 - 0.011.x_2 - 0.008.x_3 + 0.937.x_4 + 4.35$
T_5	$-0.022.x_1 - 0.017.x_2 - 0.021.x_3 + 0.925.x_4 + 5.97$
T_6	$0.014.x_1 - 0.025.x_2 - 0.03.x_3 + 0.939.x_4 + 5.15$
T_7	$0.0008.x_1 - 0.003.x_2 + 0.0153.x_3 + 0.925.x_4 + 2.93$
T_8	$0.0007.x_1 - 0.0009.x_2 - 0.005.x_3 + 0.9259.x_4 + 4.20$
T_9	$0.0029.x_1 + 0.0183.x_2 - 0.015.x_3 + 0.947.x_4 + 2$

TABLE 4.12 – Modèles de prédiction pour le temps de réponse

Tâche	Modèle débit
T_1	$-0.029.x_1 + 0.001.x_2 + 0.008.x_3 + 0.891.x_4 + 2.21$
T_2	$-0.001.x_1 + 0.009.x_2 + 0.01.x_3 + 0.892.x_4 + 1.06$
T_3	$-0.0178.x_1 - 0.034.x_2 - 0.043.x_3 + 0.925.x_4 + 2.75$
T_4	$0.005.x_1 + 0.001.x_2 - 0.005.x_3 + 0.930.x_4 + 1.11$
T_5	$-0.0239.x_1 + 0.010.x_2 - 0.005.x_3 + 0.923.x_4 + 1.53$
T_6	$0.019.x_1 - 0.047.x_2 + 0.004.x_3 + 0.840.x_4 + 2.78$
T_7	$-0.0210.x_1 - 0.0248.x_2 - 0.0185.x_3 + 0.917.x_4 + 2.27$
T_8	$-0.0086.x_1 - 0.0381.x_2 - 0.0524.x_3 + 0.915.x_4 + 3.11$
T_9	$0.0159.x_1 - 0.0639.x_2 + 0.0374.x_3 + 0.8832.x_4 + 2.30$

TABLE 4.13 – Modèles de prédiction pour le débit

Pour la prédiction des contraintes de QoS locales, nous prenons comme entrées

Algorithme 7 : Prédiction des contraintes de QoS locales

Entrées : QCB : Cas de base de QoS,
h : le nombre de cas similaires récupérés,
w_1 and w_2 : poids des distances de similarité,
t : taille de la fenêtre d'entraînement
Sorties : Contraintes de QoS locales

DÉBUT
Étape 1. Construire le cas cible des contraintes de QoS locales
$Cas_0 =< PR_0, ?LQoSCM_0 || H_0, L_0 >^{TT_0}$
Étape 2. Récupérer les cas de base similaires
pour *chaque cas historique récupéré* Cas_i **faire**
    Calculer la similarité entre le $i^{ème}$ cas et le cas cible Cas_0 :
        SIM$(Cas_0, Cas_i)\leftarrow$ Équation 4.15.
fin
Trier les cas historiques en fonction des similitudes du plus proche au plus éloigné
Sélectionner les premiers h cas
Étape 3. Construire des points de données d'entraînement
pour *chaque tâche de processus* j **faire**
    pour *chaque attribut de QoS* k **faire**
        pour z=1 *à* $h - t$ **faire**
            Construire le $z^{ème}$ exemple d'entraînement
            $(X_z, Y_z) = (\left\{e_{jk}^z, e_{jk}^{z+1}, ..., e_{jk}^{z+t-2}\right\}, e_{jk}^{z+t-1})$
        fin
    fin
fin
Étape 4. Construire un modèle de prédiction de régression SVM
Entraînez la régression SVM sur les exemples d'entraînements :
    - Obtenir la fonction de prédiction (problème dual donné en Équation 4.22)
    - Obtenir les valeurs optimales des paramètres de régression SVM : C, fonction noyau et ses paramètres
    - Résoudre le problème et obtenir α et α^*
Étape 5. Prédiction des contraintes de QoS locales
pour *chaque tâche de processus* j **faire**
    pour *chaque attribut de QoS* k **faire**
        Prédire e_{jk}^0 :
            $e_{jk}^0 = f(X)$, tel que X est l'ensemble des entrées e_{jk}^z des matrices des derniers $t-1$ cas similaires récupérés
    fin
fin
Fin

les quatre contraintes de QoS locales les plus récentes pour chaque tâche et chaque paramètre de QoS dans la base de connaissance sous la forme d'un vecteur de 4 dimensions (taille de vecteur X). En considérant les modèles construits précédemment, les contraintes locales de cas cible sont données dans le Tableau 4.14.

	Temps de réponse		Débit	
Tâche	Vecteur d'entrée	e^0_{j1}	Vecteur d'entrée	e^0_{j2}
T_1	60.66, 61.24, 62.53, 59.75	59.77	17.67, 16.58, 16.96, 16.4	16.47
T_2	46.81,46.75,45.59,42.69	42.79	12.11,11.6,11.08,11.62	11.64
T_3	56.81,57.9,58.67,56.45	56.54	16.34,17.12,14.12,15.94	16.01
T_4	46.32,47.67,45.23,46.58	46.53	14.95,15.89,14.85,14.98	15.07
T_5	44.29,45.42,43.5,45.34	45.23	14.59,15.41,17.27,15.14	15.24
T_6	50.65,48.05,50.25,52.0	51.9	15.35,15.24,15.73,14.98	15.02
T_7	48.9,46.21,46.94,48.46	48.37	16.74,14.49,15.13,14.39	14.49
T_8	52.06,52.49,51.25,53.5	53.43	16.79,16.55,17.15,16.31	16.36
T_9	43.04,41.54,41.99,40.99	41.08	18.32,18.77,18.88,18.21	18.18

TABLE 4.14 – Contraintes locales prédites

Après la prédiction des contraintes locales, nous procédons à une sélection locale au niveau de chaque cluster. Donc, les résultats finaux de la sélection locale sont : l'avatar A_{16} pour la tâche T_1, A_{13} pour T_2, A_7 pour T_3, A_7 pour T_4, A_1 pour T_5, A_{10} pour T_6, A_{16} pour T_7, A_3 pour T_8 et A_1 pour T_9.

4.5 Conclusion

Tout au long de ce chapitre, nous avons présenté nos deux dernières contributions concernant la sélection distribuée des services IoT. Contrairement aux méthodes existantes, notre proposition traite des valeurs de QoS dépendantes du temps, c'est-à-dire, la fluctuation des paramètres de QoS ce qui garantit la fiabilité de la composition résultante.

Dans notre première contribution, nous avons présenté la méthode de sélection basée sur la décomposition des contraintes de QoS globales en contraintes locales via MOEA de sorte que les avatars élus des clusters distribués sélectionnent localement l'avatar le plus adapté pour atteindre une solution quasi-optimale. Cette méthode est particulièrement significative pour les applications IoT avec un environnement dynamique à large échelle avec des exigences de QoS de bout en bout.

En ce qui concerne la deuxième contribution de ce chapitre, elle consiste en la prédiction des contraintes de QoS locales en se basant sur l'historique des expériences antérieures des avatars lors de l'achèvement de leurs objectifs complexes. La sélection basée sur la prédiction est réalisée via l'application de la méthode SVM qui fait partie des méthodes d'apprentissage automatique puissantes et évolutives.

Le travail réalisé dans le cadre de cette thèse a été implémenté et évalué en se basant sur des simulations. Le prochain chapitre sera consacré à la présentation des évaluations réalisées ainsi qu'à l'analyse et à l'interprétation des résultats obtenus. Nous nous intéressons particulièrement à l'évaluation de la performance et de l'efficacité en termes de temps de réponse et d'optimalité de nos approches.

Chapitre 5

Évaluation expérimentale

Sommaire

Ce chapitre est dédié à l'évaluation des solutions proposées. Nous décrivons tout d'abord l'architecture technique et l'environnement de réalisation de notre système et les principaux choix techniques et logiciels utilisés. Ensuite, nous présentons les résultats expérimentaux de nos propositions en trois principales parties.

La première partie est consacrée à l'évaluation de la méthode de découverte des services IoT basée sur le réseautage social et le clustering. Une étude portant, tout d'abord, sur la génération d'avatars utilisés pour l'évaluation de la découverte,la construction du réseau social, et le clustering est réalisée. Ensuite, une étude de performance globale pour l'ensemble du processus de découverte basée sur le temps de réponse et le taux de réussite est présentée. Dans la deuxième partie, la méthode de sélection distribuée des services IoT basée sur l'algorithme MOEA est évaluée. Tout d'abord, nous avons étudié les paramètres relatifs à l'algorithme génétique en terme de diversité et de convergence pour pouvoir choisir les paramètres les plus adéquats. Ensuite, le processus de sélection

est évalué en termes de temps de réponse et d'optimalité. En ce qui concerne la troisième partie, elle est destinée à évaluer l'approche de sélection basée sur la méthode de régression SVM. En premier, les paramètres de l'algorithme SVM doivent être déterminés en étudiant son comportement via une série d'expériences. Ensuite, le processus générique de la sélection est évalué via les mêmes critères utilisés pour l'algorithme MOEA pour pouvoir faire une comparaison entre eux par la suite.

5.1 Environnement d'exécution et d'implémentation

Pour ces évaluations, un ensemble d'outils et de technologies sont utilisés :

- Spring Boot [1] : pour l'implémentation et le codage des avatars, nous avons choisi la technologie Spring Boot pour les représenter sous forme de micro-services. Plusieurs autres technologies peuvent être également utilisées telles que : Restlet [2], Spark [3], et Dropwizard [4]. Le choix des micro-services Spring Boot est justifié par le fait que leurs propriétés sont adéquates avec les caractéristiques des avatars qui sont des composants indépendants, découplés, distribués et évolutifs.
- Apache HttpComponents [5] : dans nos approches de découverte et de sélection, les avatars ont besoin de communiquer et de s'envoyer des messages pour qu'ils puissent collaborer entre eux et accomplir des tâches complexes.
- XML : le format de représentation XML est choisi pour transporter les messages échangés entre les avatars pendant le processus de découverte et de sélection.
- OWL [6] : pour les bases des connaissances des avatars, i.e., l'instanciation de l'ontologie *AvatarOnt*, et leurs méta-données, le langage OWL est utilisé.
- Apache JENA [7] : pour implémenter la logique de raisonnement automatique au sein des avatars, le framework Apache JENA est utilisé.

En ce qui concerne l'environnement d'exécution des tests, nous avons utilisé un serveur Ubuntu multiprocesseur 251 GO RAM, AMD Opteron pour permettre l'exécution d'un grand nombre d'avatars et tester la scalabilité de nos approches.

5.2 Évaluation de l'approche de découverte

Avant l'évaluation de l'approche de découverte en termes de temps de calcul (computation time) et le taux de réussite (success rate), une évaluation de l'étape de génération

1. https://spring.io/projects/spring-boot
2. http://restlet.talend.com
3. http://sparkjava.com/
4. https://www.dropwizard.io/en/stable/
5. https://hc.apache.org/index.htm
6. https://www.w3.org/OWL/
7. https://jena.apache.org/

des avatars, qui précède le processus de découverte, est menée. Ensuite, une évaluation portant sur la construction du réseau social d'avatars et leur répartition en plusieurs clusters selon les fonctionnalités qu'ils fournissent est réalisée.

Enfin, une étude comparative des performances globales du processus de découverte entre notre approche et l'approche de découverte UDDI traditionnelle (qui n'utilise pas le réseautage social et le clustering) est présentée.

Nous démarrons les tests en générant les données sémantiques de chaque avatar sous forme de fichiers .owl. Après cela, nous exécutons automatiquement un nombre spécifique d'avatars implémentés sous forme de micro-services. Ensuite, nous sélectionnons un avatar initiateur pour commencer la découverte d'un objectif généré d'une manière aléatoire.

5.2.1 Génération des avatars

Pour l'initialisation de notre système, nous avons construit un générateur de données sémantiques qui permet de créer des fichiers des bases de connaissances pour chaque avatar. Ensuite, ces fichiers sont publiés et déployés dans le serveur pour des fins d'exécution.

Le graphe de la Figure 5.1 représente la variation du temps de calcul de cette étape d'initialisation en fonction du nombre d'avatars considérés. Il inclut le temps de création des fichiers sémantiques d'avatars et le temps de leur publication dans le serveur.

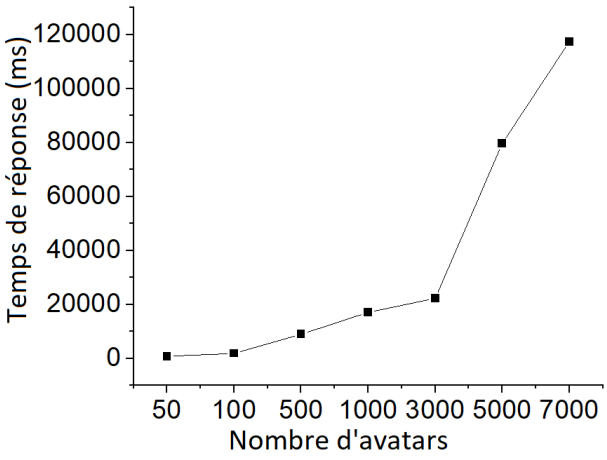


FIGURE 5.1 – Variation du temps de génération des données sémantiques en fonction du nombre d'avatars

Les résultats montrent que cette étape prend un temps de calcul considérable compte tenu de l'utilisation de descriptions sémantiques riches, mais elle n'est effec-

tuée qu'une seule fois au démarrage du système et n'aura donc pas d'impact ensuite dans la dynamique du système.

5.2.2 Construction du réseau social

Une fois que les avatars sont générés et publiés, nous procédons à l'évaluation et à l'étude de l'étape de construction du réseau social de l'avatar initiateur qui cherche à découvrir les avatars qui peuvent collaborer avec lui pour satisfaire les différentes tâches composant le processus. Il s'agit d'une étape importante de notre approche de découverte.

Le graphe de la Figure 5.2 représente la variation du temps de calcul de cette étape en fonction du nombre d'avatars considérés dans le système.

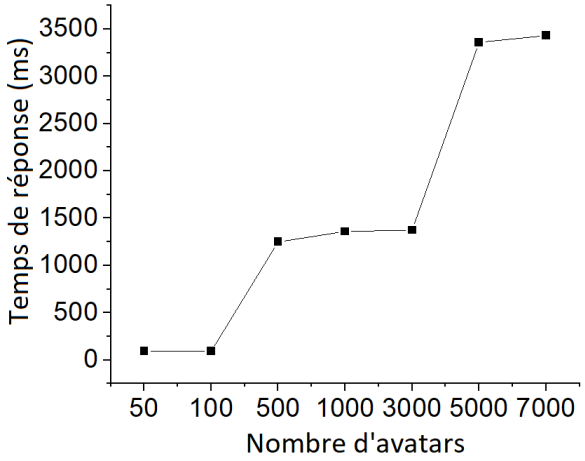


FIGURE 5.2 – Variation du temps de construction du réseau social en fonction du nombre d'avatars

Les résultats montrent que cette étape, exécutée pour chaque processus, en plus de son impact sur la réduction de l'espace de recherche de découverte, possède un temps de réponse assez réduit même pour un système à large échelle d'avatars ; par exemple, la construction d'un réseau social dans un système de 1000 avatars prend environ 1.5 secondes.

5.2.3 Clustering du réseau social

L'algorithme de clustering basé sur fuzzy c-means nécessite plusieurs paramètres d'entrée. Ces derniers incluent : la distance à utiliser pour mesurer la similarité entre

les centroïdes et les autres avatars, le paramètre de fuzzification et le critère d'arrêt epsilon. Pour choisir la configuration adéquate, nous sélectionnons quatre distances qui sont les plus utilisées [Arora et al., 2019] : la distance euclidienne, la distance euclidienne standard, la distance manhattan, et la distance camberra. Pour le paramètre de fuzzification m, nous utilisons huit valeurs dans l'intervalle [1.1,5] avec un pas de 0.5 en se basant sur les résultats fournis dans le travail de [Zhou et al., 2014]. En ce qui concerne le paramètre epsilon, nous utilisons six valeurs {0.0001, 0.001, 0.01, 0.1, 0.2, 0.3}. Le choix des valeurs de ces paramètres influence fortement la qualité de clustering et par conséquent l'approche de découverte. Pour cela, nous proposons d'utiliser la méthode de validation de clustering par indices (Clustering Validation Indices, CVI) en utilisant trois indices : l'indice XB (Xie and Beni index), l'indice PC (Partition Cluster) et l'indice PE (Partition Entropy) qui mesurent la qualité de clustering [Zhou et al., 2014] (cf. Annexe B, Section B.1).

Le graphe de la Figure 5.3 représente la variation de la qualité de clusering mesurée en utilisant la moyenne des trois indices indiqués ci-dessus (XB, PC et PE) en fonction de la distance utilisée et le paramètre de fuzzification en fixant epsilon à 0.1, le nombre d'avatars à 1000 et le nombre de tâches composant le processus à satisfaire à 10.

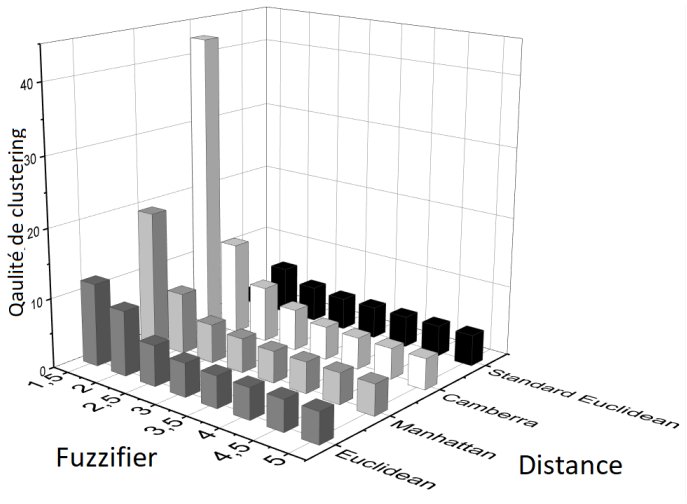


FIGURE 5.3 – Variation de la qualité du clustering en fonction de la distance et le paramètre de fuzzification

Le graphe de la Figure 5.4 représente la variation du temps de calcul de l'étape du clustering en fonction de la distance utilisée et le paramètre de fuzzification tout en fixant aussi le critère epsilon à 0.1, le nombre d'avatars à 1000 et le nombre de tâches composant le processus à satisfaire à 10.

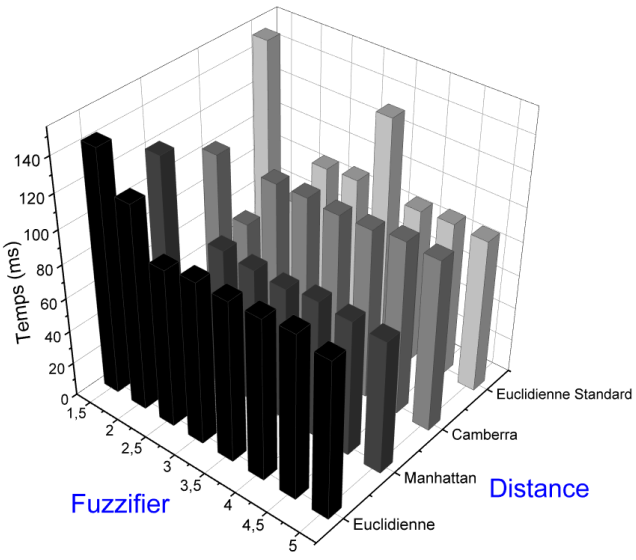


FIGURE 5.4 – Variation du temps du clustering en fonction de la distance et du paramètre de fuzzification

Le graphe de la Figure 5.5, quant à lui, représente la variation de la qualité de clustering et le temps de calcul que cette étape prend, en fonction d'epsilon en utilisant la distance euclidienne standard et en fixant le fuzzifier à 2, le nombre d'avatars à 1000 et le nombre de tâches à 10.

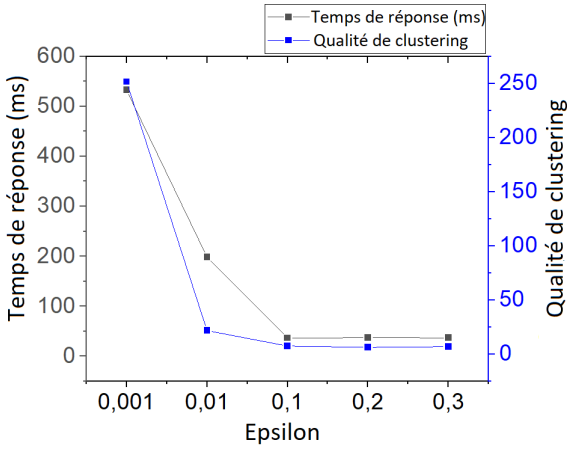


FIGURE 5.5 – Variation de la qualité et du temps de calcul de l'étape de clustering en fonction des valeurs d'epsilon

Selon les résultats fournis par les graphes des Figures 5.3, 5.4 et 5.5, nous pouvons confirmer que : la distance qui peut offrir une bonne qualité de clustering ainsi qu'un

temps de calcul réduit est la distance euclidienne standard, la valeur adéquate pour le paramètre de fuzzification est 2, et que la bonne valeur d'epsilon est 0.1.

Une fois que les paramètres de l'algorithme fuzzy c-means sont fixés, une évaluation globale de l'étape de clustering selon la configuration choisie est effectuée. Le graphe de la Figure 5.6 représente la variation de temps de calcul de l'étape de clustering en fonction du nombre d'avatars et le nombre de tâches composant le processus à satisfaire.

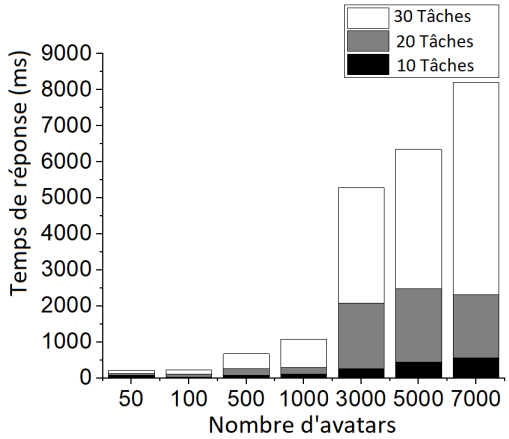


FIGURE 5.6 – Variation du temps de calcul de clustering en fonction du nombre d'avatars selon la configuration choisie

Les résultats montrent que le clustering nécessite un temps d'exécution réduit même avec un nombre d'avatars important. À titre d'exemple, le cluetring de 500 avatars en 10 clusters (selon les tâches qui composent le processus), qui est le cas le plus réaliste, ne prend qu'environ 0,1 s.

5.2.4 Étude comparative

Dans cette section, nous présentons une étude comparative entre notre approche de découverte distribuée basée sur le réseautage social et le clustering et l'approche UDDI centralisée qui exclut les étapes de clustering et la construction de réseaux sociaux.

Les graphes de la Figure 5.7 et la Figure 5.8 représentent la variation du temps de calcul et du taux de réussite des deux approches de découverte en fonction du nombre d'avatars en fixant le nombre de tâches à 12, la taille du réseau social à 40 et la configuration du clustering choisie selon les expérimentations précédentes.

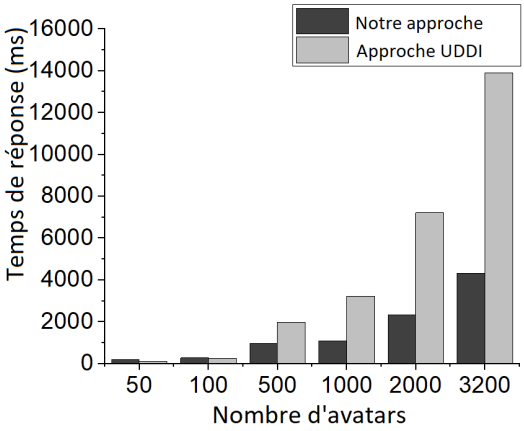


FIGURE 5.7 – Variation du temps de réponse des deux approches en fonction du nombre d'avatars

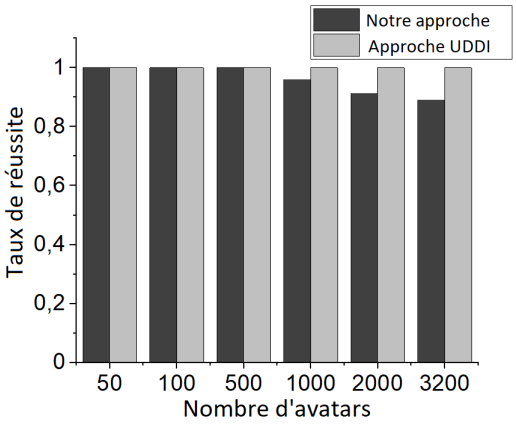


FIGURE 5.8 – Variation du taux de réussite des deux approches en fonction du nombre d'avatars

Les résultats montrent que les deux méthodes donnent approximativement les mêmes résultats concernant le temps de réponse pour un petit réseau d'avatars (moins de 500 avatars) tout en garantissant le même taux de réussite qui est égal à 1, i.e., la découverte des avatars requis est accomplie à 100%. Cependant, pour les réseaux de grande taille (plus de 500 avatars), notre approche surpasse en terme de temps d'exécution l'approche centralisée tout en garantissant un taux de réussite élevé. En effet, contrairement à l'approche centralisée, notre approche de découverte distribuée n'explore pas tous les avatars du système tout en gardant un taux de réussite supérieur à 83%.

5.3 Évaluation de la méthode de décomposition des contraintes de QoS globales : MOEA

Dans cette section, nous étudions la complexité de l'approche basée MOEA. Aussi, cette approche est évaluée expérimentalement en terme de temps de calcul et d'optimalité. Enfin, une étude comparative de l'approche que nous proposons avec l'approche de sélection globale est donnée.

5.3.1 Évaluation de la complexité

L'évaluation de la complexité de notre approche de sélection locale dépend des paramètres suivants : 1) le nombre des clusters n, 2) le nombre d'avatars par cluster l, 3) le nombre de contraintes de QoS globales considérées r, et 3) le nombre de niveaux

de qualité d. Notre approche de sélection comprend les étapes suivantes : le calcul des niveaux de qualité, l'algorithme MOEA pour la décomposition des contraintes de QoS globales, et la sélection locale d'un avatar de chaque cluster.

Par conséquent, pour le calcul des niveaux de qualité et la sélection locale, la complexité est linéaire et ne dépend que du nombre d'avatars par cluster l, car chacun des élus des clusters effectue ces étapes localement et en parallèle. Donc la complexité de ces deux étapes est $O(2l)$. Concernant l'algorithme MOEA pour la décomposition des contraintes de QoS globales et comme il s'agit d'un algorithme évolutionnaire, sa complexité est également linéaire et dépend du nombre de générations g, de la taille de la population P_{size} et du nombre de variables du problème $n.r.d$. Donc, la complexité de cette étape est $O(n.r.d.g.P_{size})$.

5.3.2 DataSet

Pour l'évaluation et la validation de notre approche de sélection, nous nous sommes basés sur le dataset fourni dans [White et al., 2018]. Ce dernier est une adaptation dans le contexte de l'IoT de l'ensemble des données des services Web WS-DREAM du dataset proposé dans [Zheng et al., 2014]. Il se compose d'une multi-matrice du temps de réponse et du débit, de 339 utilisateurs pour 5 825 services IoT. Nous rappelons que le temps de réponse représente le temps entre l'envoi de la requête et la réception d'une réponse et le débit indique le taux de transmission des données. Ce dataset offre un grand nombre d'observations (1 974 675 observations), ce qui assure une évaluation pertinente de notre approche. En plus, la variation dans le temps des attributs de QoS provoquée par la congestion du réseau permet de prendre en compte la fluctuation des paramètres de QoS.

Pour exploiter ce dataset, nous choisissons des services aléatoires et des utilisateurs aléatoires pour constituer les clusters d'avatars. Concernant les données historiques utilisées pour le calcul de la fluctuation des paramètres de QoS, nous utilisons les données des mêmes services choisis avec des observations différentes (il y a 339 observations pour chaque service).

5.3.3 Résultats expérimentaux

Nous commençons notre étude expérimentale en choisissant les paramètres adéquats de l'algorithme MOEA en fonction de la convergence et de la diversité de la population finale obtenue. Pour cela, deux indices sont utilisés : l'indice de distance générationnelle inversée (Inverted Generational Distance, IGD) pour mesurer la convergence de la population et l'indice d'hyper-volume (Hyper-volume, HV) pour mesurer sa diversité (cf. Annexe B, Section B.2). Une petite valeur de l'indice IGD et une valeur élevée de l'indice HV indiquent une meilleure convergence et une meilleure diversité respectivement. Ces deux indices sont combinés dans un indice global nommé indice de qualité

(Quality Index, QI) défini comme suit : $QI = HV + \frac{1}{IGD}$, tel que : une plus grande valeur de QI dénote une meilleure qualité.

5.3.3.1 Paramètres d'évolution de la population

Pour déterminer les paramètres adéquats d'évolution de la population, nous considérons une instance de problème de grande taille avec 10 clusters, 1000 avatars par cluster et 20 niveaux de qualité. Par conséquent, la taille de l'espace de recherche est : $d^{(r*n)} = 20^{2*10}$.

Le graphe de la Figure 5.9 représente la variation de la valeur de l'indice de qualité (QI) en fonction de la taille de la population et de la taille du voisinage.

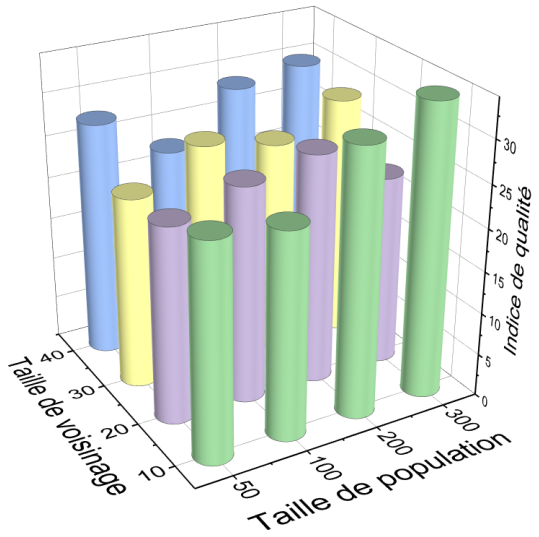


FIGURE 5.9 – Variation de la valeur de QI en fonction de la taille de la population et de la taille de voisinage

Les résultats montrent que les valeurs 300 et 10 semblent être les meilleures valeurs à sélectionner respectivement pour la taille de la population et la taille du voisinage, car la valeur de l'indice QI est maximisée pour ces valeurs. Cela est justifié par le fait qu'une taille suffisante de la population permet d'atteindre un nombre considérable d'autres solutions par croisement et mutation et une petite taille de voisinage permet d'avoir une couverture suffisante de l'espace de recherche via les différents sous-problèmes qui cherchent à améliorer leur meilleure solution.

La Figure 5.10 présente la variation de la valeur de l'indice de qualité (QI) en fonction du nombre de générations considérées et du nombre de solutions remplacées.

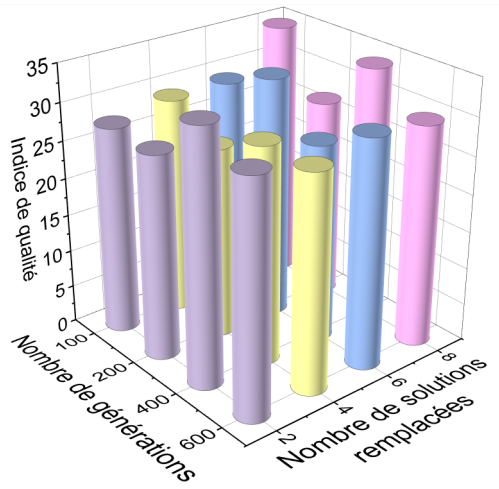


FIGURE 5.10 – Variation de la valeur de QI en fonction du nombre de générations et du nombre de solutions remplacées

Les résultats indiquent que les valeurs adéquates pour le nombre de générations et le nombre de solutions remplacées sont respectivement 400 générations et 8 individus. Ces valeurs garantissent une meilleure convergence et diversité de l'algorithme MOEA. Ceci parce que nous considérons une grande instance de problème, il a donc fallu un grand nombre de générations et un nombre important de sous-problèmes à mettre à jour à chaque itération pour garantir la convergence et la diversité respectivement.

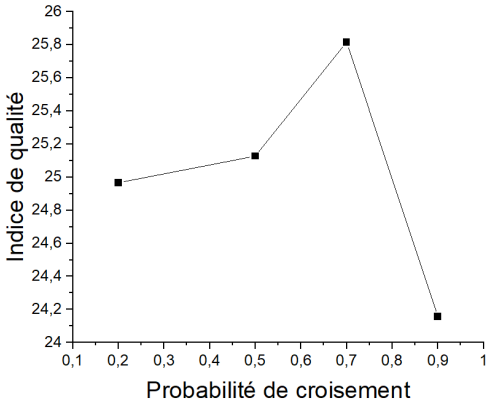


FIGURE 5.11 – Variation de la valeur de QI en fonction de la probabilité de croisement

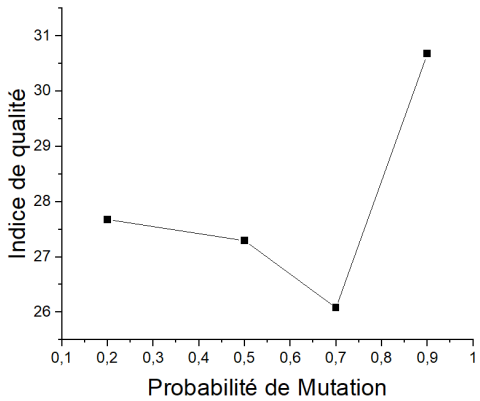


FIGURE 5.12 – Variation de la valeur de QI en fonction de la probabilité de mutation

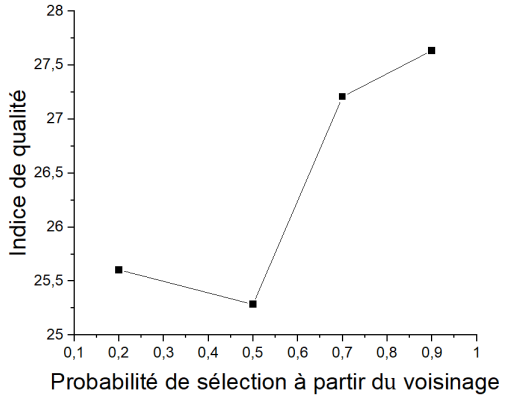


FIGURE 5.13 – Variation de la valeur de QI en fonction de la probabilité de sélection

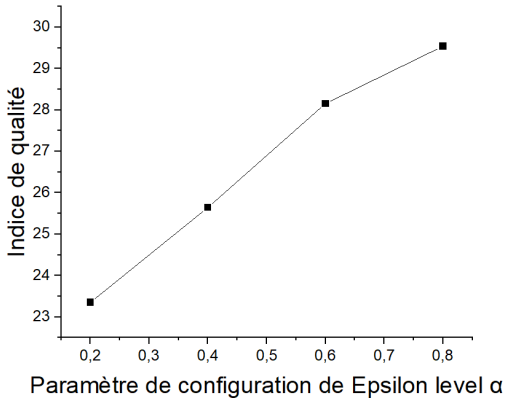


FIGURE 5.14 – Variation de la valeur de QI en fonction du paramètre de réglage du niveau epsilon α

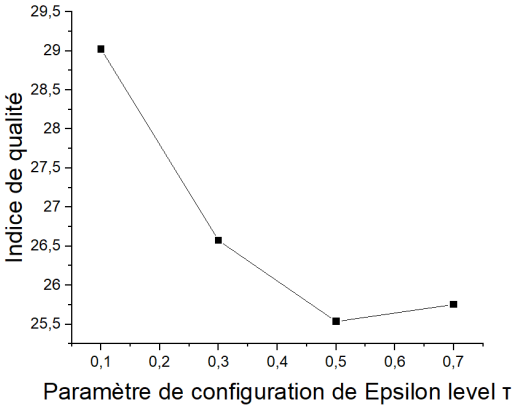


FIGURE 5.15 – Variation de la valeur de QI en fonction du paramètre de réglage du niveau epsilon τ

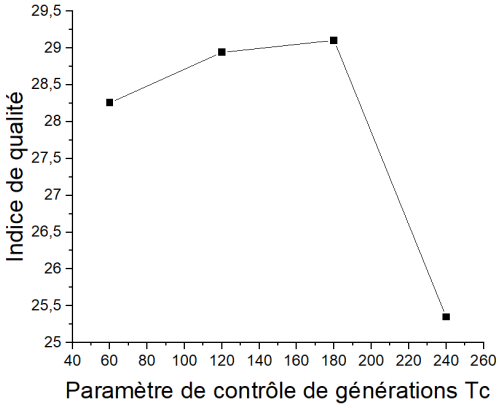


FIGURE 5.16 – Variation de la valeur de QI en fonction du paramètre de contrôle de génération

Les graphes de la Figure 5.11, de la Figure 5.12 et de la Figure 5.13 illustrent la variation de l'indice de qualité (QI) en fonction des probabilités des opérateurs d'évolution : la probabilité de croisement, la probabilité de mutation, et la probabilité de sélection respectivement.

Les résultats montrent que les valeurs adéquates pour la probabilité de croisement et de mutation sont respectivement de 0,7 et 0,9. Cela se justifie par le fait que les fortes

probabilités de croisement et de mutation ont un impact positif sur la convergence et la diversité de la population actuelle. Concernant la probabilité de sélection dans le voisinage, la valeur 0,9 est la meilleure valeur à choisir, car le voisinage des sous-problèmes permet l'échange d'informations entre les sous-problèmes pour une meilleure convergence.

Pour les graphes de la Figure 5.14, de la Figure 5.15 et de la Figure 5.16, ils représentent la variation de la valeur de l'indice de qualité (QI) en fonction des paramètres de réglage du niveau epsilon.

L'étude empirique montre que les paramètres de réglage d'epsilon, qui reflètent la préférence de la recherche des solutions dans les régions irréalisables et pour lesquels on obtient une meilleure diversité et convergence sont : α à 0,8, τ à 0,1 et le contrôle de génération à 180 itérations. Cela signifie que pour obtenir une meilleure convergence et diversité pour une instance de problème de grande taille, nous devrions considérer $(1-\alpha)$ qui est égale à 0,2 comme probabilité de recherche dans la région des solutions irréalisables avec la vitesse de relaxation des contraintes τ égale à 0,1 et ceci pour les 180 premières itérations.

5.3.3.2 Étude comparative

Après avoir fixé les paramètres de l'algorithme MOEA, une étude comparative est menée pour le comparer avec l'algorithme génétique traditionnel. Ensuite, une étude pour comparer notre approche de sélection locale, basée sur la décomposition des contraintes de QoS globales en contraintes locales en utilisant MOEA, à deux autres approches en termes de temps de calcul et d'optimalité est réalisée. La première approche utilisée est également une approche de sélection locale qui utilise le solveur vOptSolver [8] pour la décomposition des contraintes de QoS globales au lieu de MOEA. Concernant la deuxième approche utilisée, il s'agit de la méthode de sélection globale exhaustive.

1) MOEA Vs Génétique traditionnel

Notre algorithme MOEA est comparé aux algorithmes génétiques traditionnels, qui n'utilisent pas la technique de la gestion des contraintes via un epsilon pour rechercher des solutions dans les régions des solutions non-réalisables, en terme de qualité de temps de réponse en fonction de la taille de la population dans les graphes 5.17 et 5.18 respectivement.

8. https://github.com/vOptSolver

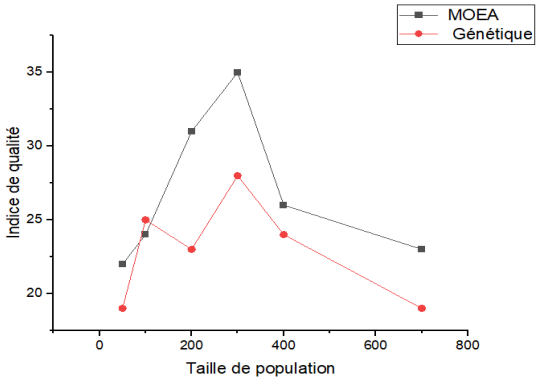


FIGURE 5.17 – Variation de l'indice de qualité en fonction de la taille de la population des approches MOEA et génétique

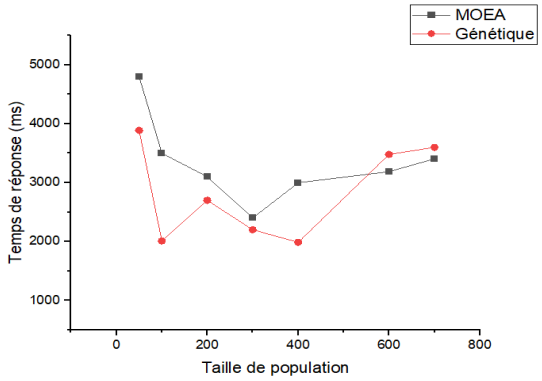


FIGURE 5.18 – Variation du temps de réponse en fonction de la taille de la population des approches MOEA et génétique

Les résultats montrent que l'algorithme MOEA donne une meilleure qualité en termes de diversité et convergence parce qu'elle exploite des régions non exploitées par les algorithmes génétiques traditionnels, et cela, avec un temps de réponse acceptable.

2) MOEA Vs vOptSolver

Les méthodes MOEA et vOptSolver sont toutes les deux basées sur la décomposition des contraintes de QoS globales, et elles ne diffèrent que dans la manière de résoudre le problème d'optimisation pour obtenir des contraintes locales. vOptSolver est une méthode de résolution exacte qui permet de choisir le schéma des contraintes locales optimal à partir des niveaux de qualité candidats, tandis que, MOEA est une méthode de résolution métaheuristique approximative.

Le graphe de la Figure 5.19 permet d'évaluer l'optimalité de l'étape de décomposition des contraintes globales de QoS de l'approche basée sur MOAE par rapport à la méthode exacte vOptSolver. L'optimalité de schéma de contraintes locales obtenu après la décomposition est définie en Équation 5.1.

$$SO_{me} = \frac{U_{MOEA}}{U_{Exact}} * 100 \tag{5.1}$$

Tels que : U_{MOEA} et U_{Exact} sont les utilités des schémas des contraintes locales obtenues par l'approche basée sur le MOEA et l'approche basée sur la décomposition exacte via vOptSolver respectivement. L'approche basée sur la décomposition exacte offre une décomposition optimale, i.e., son optimalité est égale à 100%.

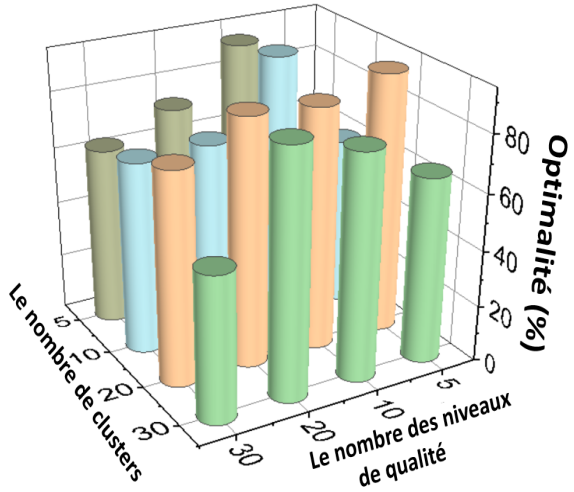


FIGURE 5.19 – Variation de l'optimalité de MOEA et de vOptSolver selon le nombre des clusters et le nombre des niveaux de qualité

À travers ce graphe, nous constatons qu'il y a une corrélation entre le nombre des niveaux de qualité et le nombre de clusters. Ces deux paramètres affectent l'optimalité de la solution trouvée. Les résultats montrent que notre approche de sélection, basée sur la décomposition des contraintes de QoS globales MOEA, permet d'obtenir un schéma de contraintes de QoS locales quasi-optimales, avec une optimalité entre [80%, 97%], pour un nombre de niveaux de qualité égale à environ 10% du nombre d'avatars par cluster.

Les graphes de la Figure 5.20 et de la Figure 5.21 représentent l'évaluation comparative entre notre approche et l'approche basée sur la méthode exacte selon le temps de réponse en fonction du nombre de clusters et du nombre des niveaux de qualité respectivement.

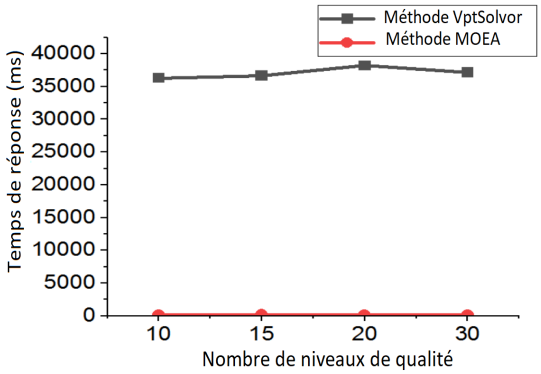


FIGURE 5.20 – Variation du temps de calcul en fonction du nombre de clusters des approches MOEA et VptSolvor

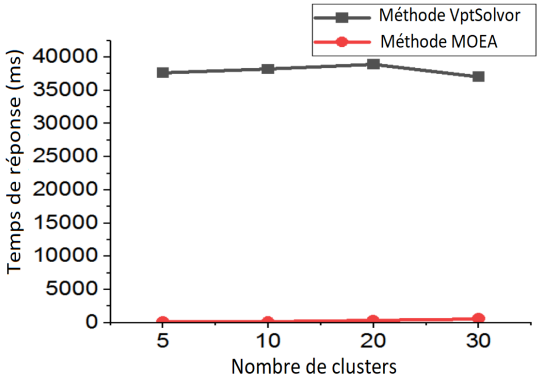


FIGURE 5.21 – Variation du temps de calcul en fonction du nombre de niveaux de qualité des approches MOEA et exacte

Les résultats montrent que la méthode de sélection basée sur l'algorithme MOEA est environ 116 fois plus rapide que celle basée sur le solveur, et cela, avec de bons résultats d'optimalité. Cela est dû au fait que la méthode exacte basée sur vOptSolver explore tout l'espace de recherche, c'est-à-dire, elle vérifie tous les schémas de contraintes locales possibles à partir des niveaux de qualité candidats, tandis que MOEA, permet une recherche réduite, mais plus ciblée.

3) MOEA Vs sélection globale

Cette partie est dédiée à l'évaluation de l'ensemble de la démarche de sélection proposée entre les deux approches par décomposition (MOEA et vOptSolver) et la méthode de sélection globale exhaustive en terme d'optimalité et du temps de calcul.

Les graphes de la Figure 5.22 et de la Figure 5.23 représentent l'évaluation des approches de sélection basées sur la décomposition qui utilisent MOEA et vOptSolver en terme d'optimalité par rapport à la méthode de sélection globale (dont l'optimalité est égale à 100%).

L'optimalité de l'étape de sélection pour le MOEA est définie comme indiqué en Équation 5.2.

$$SO_{mg} = \frac{U_{MOEA}}{U_{global}} * 100 \tag{5.2}$$

L'optimalité de l'étape de sélection pour vOptSolver est définie comme indiqué en Équation 5.3.

$$SO_{eg} = \frac{U_{exact}}{U_{global}} * 100 \tag{5.3}$$

Tels que : U_{global}, U_{MOEA} et U_{exact} sont les utilités de la solution de composition obtenue par la méthode de sélection globale, de la méthode basée sur MOEA et la méthode basée sur vOptSolver respectivement.

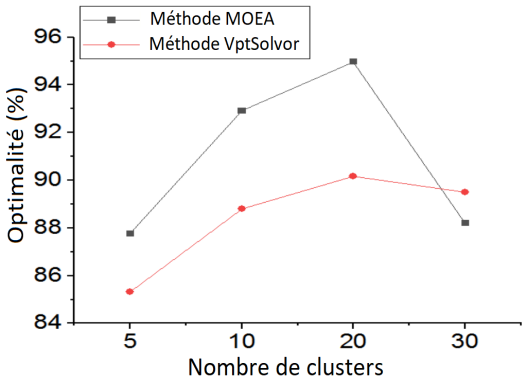


FIGURE 5.22 – Variation de l'optimalité de sélection basée sur MOEA et vOptSolver par rapport à la sélection globale en fonction du nombre de clusters

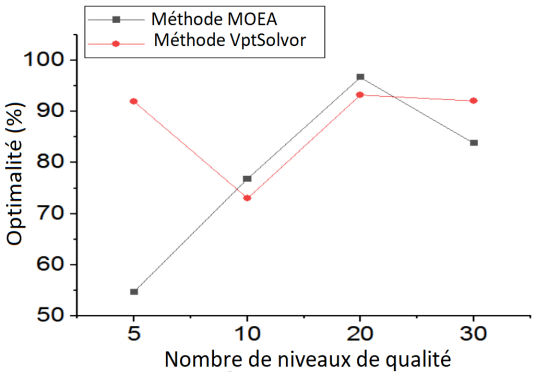


FIGURE 5.23 – Variation de l'optimalité de sélection basée sur MOEA et vOptSolver par rapport à la sélection globale en fonction du nombre de niveaux de qualité

Les résultats montrent tout d'abord que notre approche de sélection basée sur MOEA offre de meilleurs résultats par rapport à la méthode basée sur vOptSolver en terme d'optimalité de la composition sélectionnée. Cela est dû au fait que l'approche de décomposition exacte basée sur vOptSolver sélectionne des contraintes locales très restrictives pendant le processus de décomposition des contraintes de QoS globales. Par conséquent, de nombreux clusters sont trouvés avec très peu de propositions d'avatars ce qui a un impact négatif sur l'utilité de la solution de composition. Ils montrent également que les méthodes de sélection locale basées sur la décomposition des contraintes de QoS globales, qu'elles soient MOEA ou exacte, offrent une optimalité (entre [55%, 97%]).

Les graphes donnés dans la Figure 5.24 et la Figure 5.25 représentent une évaluation du temps de calcul entre les trois approches : notre approche MOEA, la méthode de décomposition exacte basée sur vOptSolver et l'approche de sélection globale. Cette étude est réalisée en fonction du nombre de clusters et du nombre d'avatars par cluster respectivement. L'évaluation en fonction du nombre des niveaux de qualité n'est pas prise en compte, car la sélection globale est indépendante de ce paramètre.

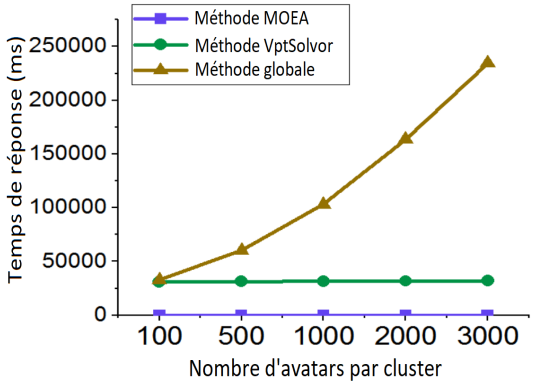


FIGURE 5.24 – Variation du temps de réponse en fonction du nombre d'avatars par cluster

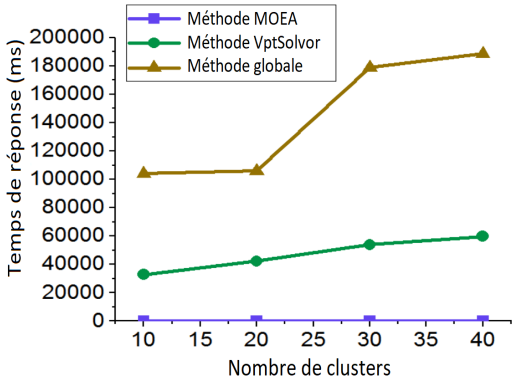


FIGURE 5.25 – Variation du temps de réponse en fonction du nombre de clusters

Les résultats démontrent que notre approche de sélection locale basée sur la décomposition des contraintes de QoS globale en utilisant MOEA est extrêmement efficace en terme de temps de calcul par rapport aux deux autres approches. Ceci est en particulier important dans des systèmes larges.

5.4 Évaluation de la méthode de prédiction des contraintes de QoS locales : SVM

Dans cette section, nous évaluons notre approche de sélection basée sur la prédiction des contraintes de QoS locales. Pour cela, nous étudions d'abord les résultats de l'application de la méthode de recherche par grille (Grid Search) qui permet de choisir les bonnes valeurs d'hyperparamètres de notre modèle de prédiction SVM. Ensuite, une étude expérimentale permet de choisir la valeur la plus adéquate de la fenêtre glissante utilisée pour la construction des exemples d'entraînement est réalisée. Après cela, nous validons l'ensemble de l'approche de sélection des services IoT par prédiction des contraintes de QoS locales à travers une étude expérimentale basée sur le temps de calcul et le critère d'optimalité.

5.4.1 Hyperparamètres du modèle de prédiction SVM

La recherche des hyperparamètres inclut : la constante C, la fonction Kernel $K(X, Y)$ et epsilon ε. Ces derniers seront utilisés par l'algorithme d'optimisation pour déterminer α et α^*, et par conséquent trouver le paramètre b de la fonction de prédiction. Les hyperparamètres ont un impact important sur les résultats finaux

de la prédiction. Pour choisir la meilleure configuration d'hyperparamètres, une méthode de recherche par grille est effectuée. Elle permet d'automatiser la sélection des hyperparamètres pour une méthode d'apprentissage automatique basée sur les résultats d'évaluation de validation croisée pour chaque combinaison des hyperparamètres. Dans notre étude, nous utilisons la validation croisée K-Fold, avec $K = 5$ parce que cette valeur est la plus utilisée dans les études pratiques et littéraires [Wang et al., 2015][Smets et al., 2007] [9] [10]. Nous sélectionnons également des valeurs de chaque hyperparamètre pour la recherche par grille comme suivant : la fonction Kernel dans l'ensemble {linéaire, polynomial, RBF}, la constante C dans {1.5,3,5,10} et epsilon dans {0,1,0.2,0.3, 0.5}, tels que : le coefficient et le degré du noyau polynomial sont fixés à 0 et 3 respectivement, et Gamma pour le noyau RBF est mis à $\frac{1}{F}$ où F est la taille de vecteur d'exemple d'entraînement, i.e., la taille de la fenêtre glissante dans notre cas. La métrique utilisée pour l'évaluation de la validation croisée est appelée ρ^2 (cf. Annexe B, Section B.3). La moyenne de ρ^2 pour toutes les validations croisées effectuées est calculée, tel que : une valeur moyenne proche de 1 représente de meilleures performances du modèle de prédiction.

Le graphe de la Figure 5.26 représente une visualisation des résultats de la recherche par grille et la variation de la moyenne de ρ^2 en fonction de la fonction kernel pour les différentes configurations de C et epsilon.

9. https ://datascience.stackexchange.com/questions/28158/how-to-calculate-the-fold-number-k-fold-in-cross -validation

10. http ://www.sthda.com/english/articles/38-regression-model-validation/157-cross-validation-essentials-in-r/

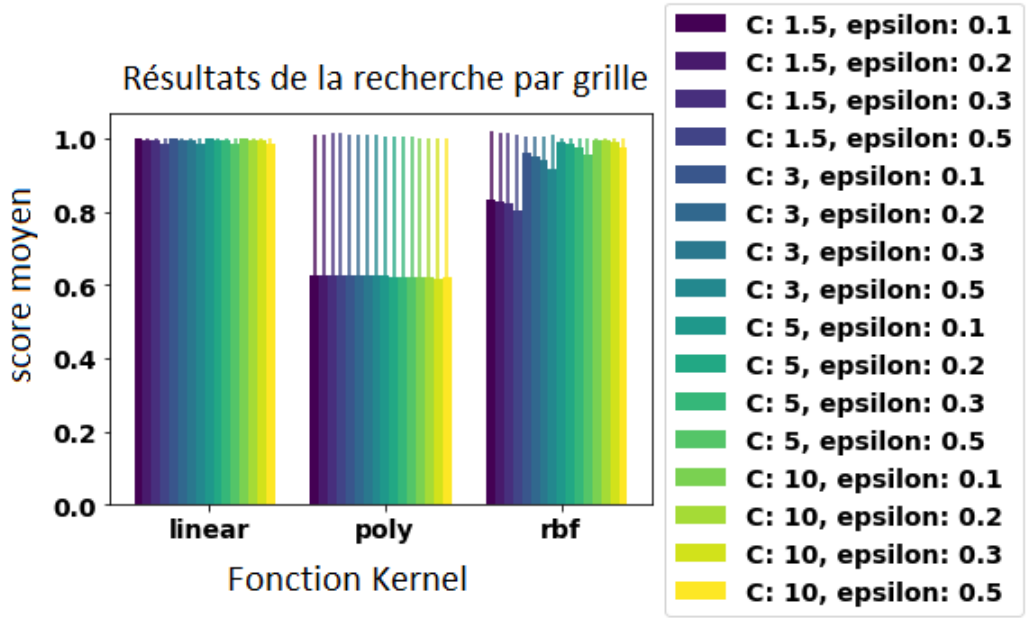


FIGURE 5.26 – Variation de la moyenne de ρ^2 en fonction de la fonction Kernel

Les résultats montrent que l'utilisation de la fonction linéaire comme une fonction kernel donne le meilleur score pour les différentes valeurs de C et epsilon utilisées.

Le graphe de la Figure 5.27 représente la variation du score de la moyenne de ρ^2 selon les valeurs de la constante C.

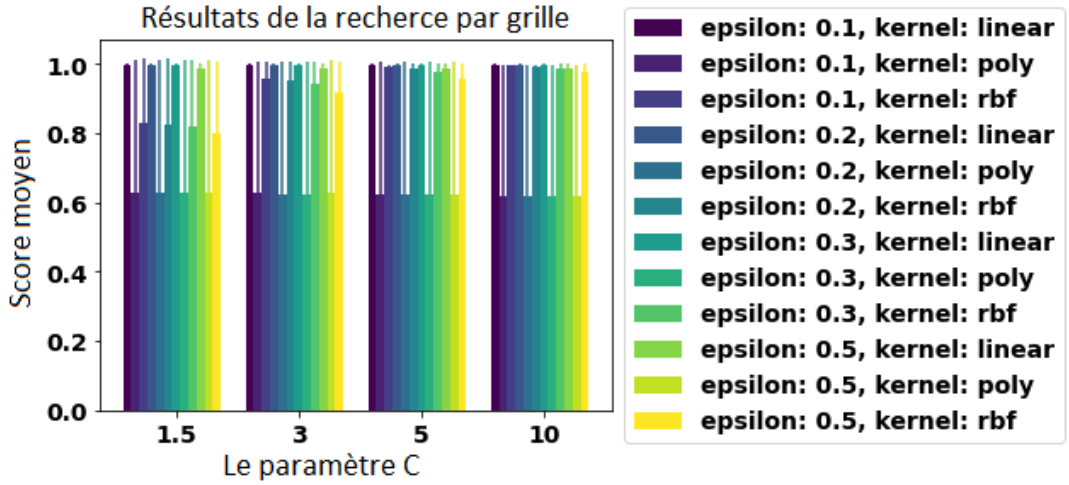


FIGURE 5.27 – Variation de la moyenne de ρ^2 en fonction du paramètre C

Les résultats montrent que 10 est la meilleure valeur de la constante C vu qu'elle donne les meilleurs scores pour les différentes valeurs d'epsilon de la fonction kernel.

Le graphe de la Figure 5.28 donne la variation de la moyenne de ρ^2 en fonction d'epsilon selon les différentes valeurs de la constante C et les fonctions kernel.

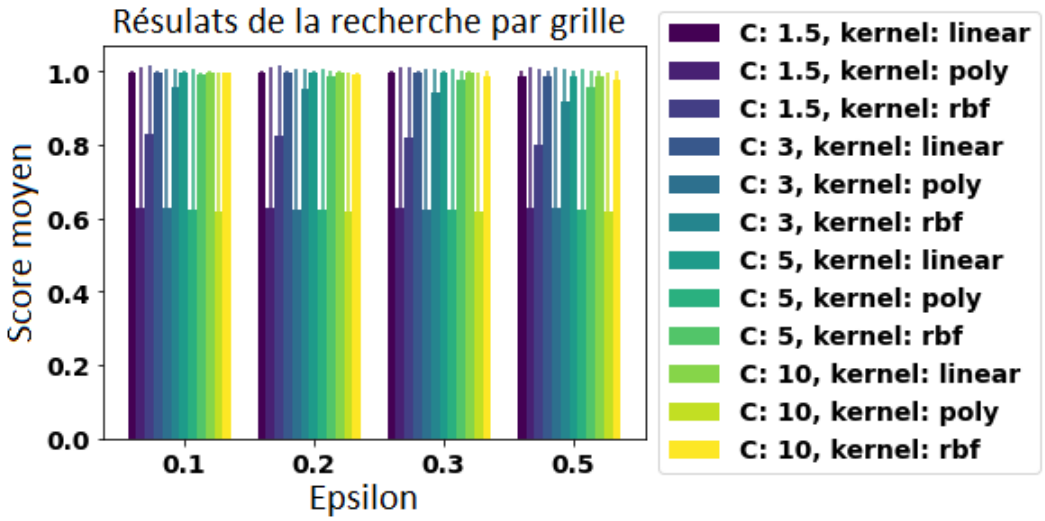


FIGURE 5.28 – Variation de la moyenne de ρ^2 en fonction d'epsilon

Les résultats de la recherche de grille ont indiqué que 0,1 est la meilleure valeur à utiliser pour le seuil epsilon.

5.4.2 Taille de la fenêtre glissante

Dans cette section, nous nous intéressons à l'étude de l'impact de la taille de la fenêtre sur les performances du modèle de prédiction SVM, et ce, en considérant les deux métriques : l'erreur quadratique moyenne (Mean squared error, MSE) et le coefficient de corrélation quadratique ρ^2 (cf. Annexe B, Section B.3). Une valeur MSE proche de 0 et une valeur ρ^2 proche de 1 correspondent aux meilleures performances. Dans ces expérimentations, nous considérons que le nombre de cas de base h récupérés de l'étape de génération de cas CBR est égal à 1000 cas.

Le graphe de la Figure 5.4.2 représente la variation des deux métriques (MSE et ρ^2) en fonction de la taille de la fenêtre glissante t utilisée pour la construction des exemples et des points d'apprentissage du modèle de prédiction.

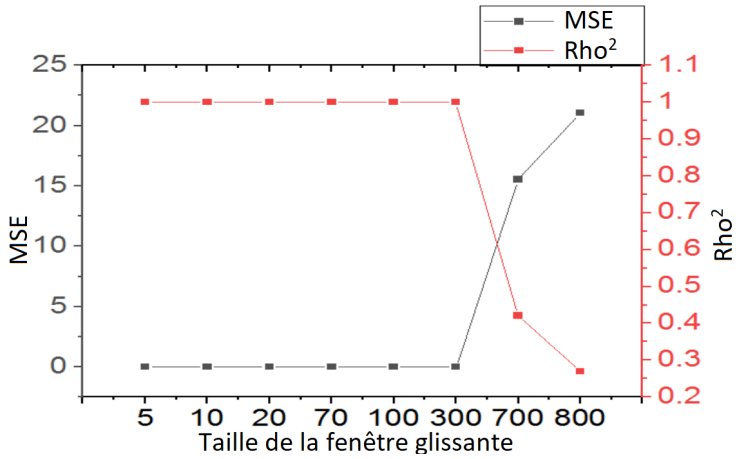


FIGURE 5.29 – Variation des métriques MSE et ρ^2 pour 1000 cas historiques en fonction de la taille de la fenêtre glissante

Les résultats montrent que la meilleure taille de fenêtre est celle qui permet de générer suffisamment de données pour l'apprentissage. Donc, d'après le graphe, pour une meilleure prédiction, la taille de la fenêtre doit être inférieure à 30% du nombre des cas de base similaires h, i.e. $t \leq 30\%h$.

5.4.3 Évaluation de la sélection par prédiction SVM

Dans cette partie, nous présentons d'abord une étude d'évaluation basée sur l'optimalité et le temps de calcul de l'approche de sélection locale basée sur la prédiction des contraintes de QoS locales pour chaque tâche composant le processus à exécuter, et cela, pour chaque paramètre de QoS. Ensuite, nous exposons une étude comparative avec une approche de sélection globale centralisée et deux autres approches distribuées basées sur la décomposition des contraintes de QoS globales : la méthode de décomposition exacte basée sur VptSolvor et la méthode basée sur MOEA.

Le graphe de la Figure 5.30 représente la variation du temps de calcul et de l'optimalité (par rapport à la méthode de sélection globale centralisée) de l'approche de sélection par prédiction SVM en fonction du nombre de tâches composant le processus à exécuter pour lesquelles nous cherchons à prédire les contraintes de QoS locales (c'est également le nombre de clusters pour les méthodes basées sur VptSolvor et MOEA).

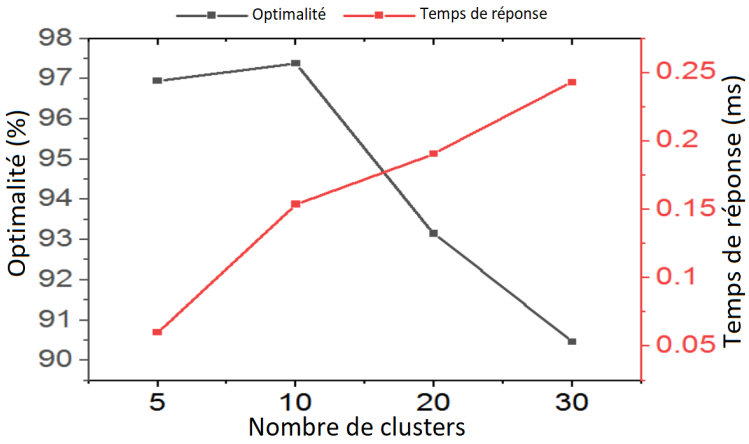


FIGURE 5.30 – Variation du temps et de l'optimalité de la méthode de sélection basée sur la prédiction SVM en fonction du nombre de clusters

Les résultats montrent que l'approche proposée offre une bonne optimalité (entre [90%, 97,5%]) et cela avec des valeurs de temps de calcul négligeables (0,25 ms pour 30 clusters).

Le graphe de la Figure 5.31 représente une évaluation comparative en terme du temps de réponse selon le nombre de clusters de notre approche de sélection basée sur la prédiction SVM par rapport à la méthode de sélection globale et les deux méthodes de sélection basées sur la décomposition des contraintes de QoS globales qui utilisent la décomposition exacte par VptSolvor et la décomposition approximative par MOEA.

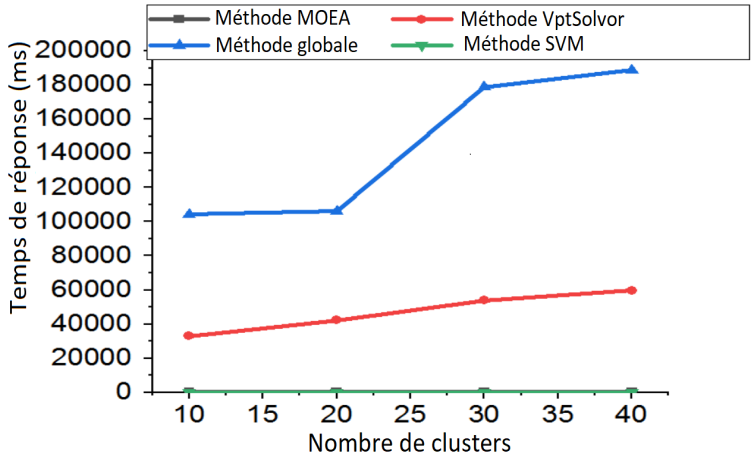


FIGURE 5.31 – Variation du temps de calcul en fonction du nombre de clusters pour les méthode SVM, MOEA et VptSolvor

Le graphe de la Figure 5.32 représente un zoom sur la partie comparative entre l'approche de sélection par prédiction SVM et l'approche de sélection basée sur MOEA afin de visualiser la différence entre elles.

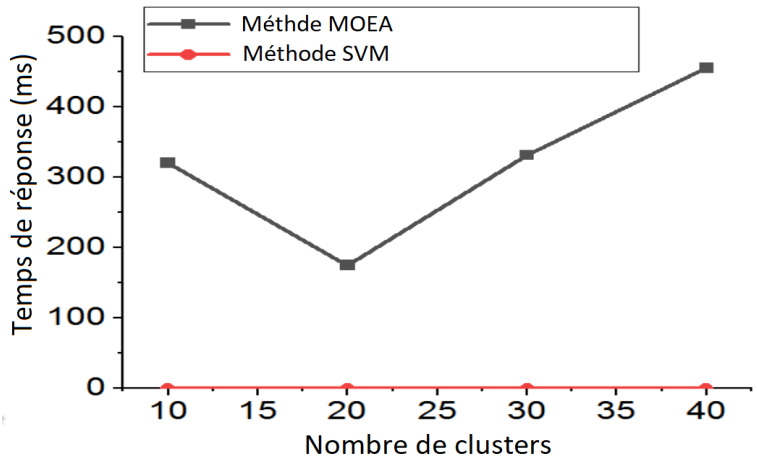


FIGURE 5.32 – Variation du temps de calcul en fonction du nombre de clusters pour la sélection basée sur la prédiction SVM et celle basée sur MOEA

Les résultats montrent que l'approche de sélection basée sur la prédiction SVM est la meilleure en ce qui concerne le temps de calcul. Cela est justifié par le fait que l'opération de prédiction ne nécessite pas beaucoup de temps (pour notre cas 0,2 ms pour 30 clusters), car cette méthode s'affranchit de l'étape de collaboration et de communication entre l'initiateur et les élus. Par conséquent, cela permet d'obtenir de bons résultats pour le temps de calcul.

Le graphe de la Figure 5.33 représente une évaluation comparative entre l'approche de sélection par prédiction SVM en terme d'optimalité en fonction du nombre de clusters par rapport aux méthodes de sélection basées sur la décomposition des contraintes de QoS globales : décomposition exacte par VptSolvor et par MOEA, et cela, au regard de la méthode de sélection globale centralité exhaustive dont l'optimalité est 100%.

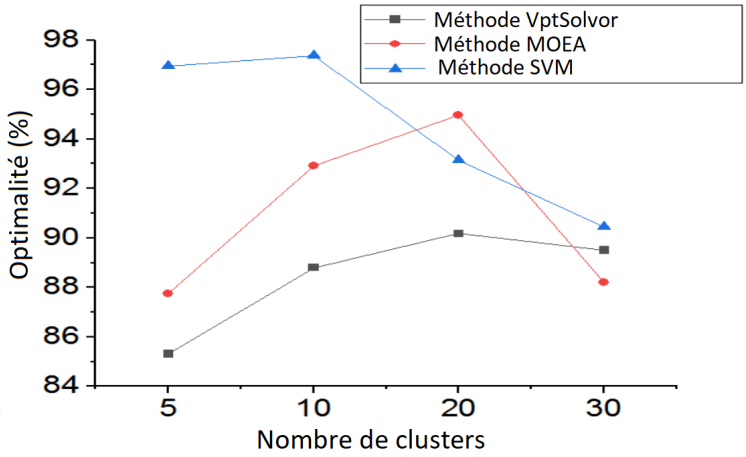


FIGURE 5.33 – Variation de l'optimalité en fonction du nombre de clusters pour les approches SVM, MOEA et VptSolvor

L'étude comparative basée sur l'optimalité montre que l'approche de sélection basée sur la prédiction SVM est la meilleure méthode de sélection par rapport aux autres approches étudiées. Son optimalité est située dans l'intervalle [90%,98%]. Cela est justifié par le fait que l'approche de sélection basée sur la prédiction SVM a appris les contraintes de QoS locales des expériences précédentes. Pour ces expérimentations, nous avons utilisé les résultats retournés par l'approche de sélection basée sur MOEA qui donne également de très bons résultats en terme d'optimalité comme données historiques pour l'apprentissage.

5.5 Conclusion

Nous avons présenté à travers ce dernier chapitre les différents tests que nous avons menés pour évaluer nos solutions relatives à la découverte et la sélection distribuée des services IoT afin d'évaluer leurs performances en termes de qualité des résultats retournés et le temps de réponse.

Tout d'abord, nous avons commencé par la présentation de l'environnement technique et des outils logiciels que nous avons utilisés pour la mise en œuvre de nos propositions. Ensuite, nous avons présenté l'évaluation des solutions proposées dans les chapitres précédents qui concerne la phase de découverte à base de réseautage social et de clustering, et la phase de sélection réalisée via l'algorithme MOEA et la méthode SVM.

À l'issue des tests réalisés durant la phase de découverte, nous avons pu montrer que l'adoption des concepts de réseautage social dans la recherche de services IoT permet d'accélérer la découverte en minimisant l'espace de recherche uniquement aux

avatars les plus aptes à satisfaire les services recherchés. En plus, le partitionnement de cet espace de recherche en plusieurs groupes selon les fonctionnalités des avatars permet d'orienter au mieux l'acheminement et la redirection des requêtes. Les résultats montrent que notre approche surpasse l'approche centralisée traditionnelle et elle est environ 4 fois plus rapide en particulier lorsque le nombre d'appareils est assez grand, et cela, avec un taux de réussite proche de 1.

Concernant la qualité des résultats retournés par l'algorithme MOEA pour la sélection des services basée sur la décomposition des contraintes de QoS globales en contraintes locales, les résultats expérimentaux montrent une amélioration significative en termes de temps de calcul avec des valeurs d'optimalité correctes par rapport à la méthode de sélection globale exhaustive traditionnelle.

Comme la méthode de sélection basée sur MOEA est réalisée en prenant en compte les avatars candidats au moment de la réalisation d'une application IoT, son temps d'exécution est relativement élevé. Pour cela, nous avons examiné l'utilisation d'une méthode d'apprentissage supervisé (SVM) pour la prédiction des contraintes de QoS locales à partir des expériences précédentes des avatars. Les résultats montrent que cette méthode est beaucoup plus rapide (250 fois plus rapide) que l'approche basée sur MOEA.

Dans le chapitre suivant, une conclusion de la thèse est donnée avec un résumé des contributions des chapitres précédents. En plus, une discussion sur les travaux futurs à court et à long terme est donnée.

Conclusion & Perspectives

Conclusion générale

Le développement massif de l'Internet des Objets dans de nombreux domaines et la mise à disposition des capacités de ces objets contribuent à développer de nouveaux usages. Le grand nombre d'appareils hétérogènes, distribués et mobiles rend la réalisation d'une application IoT, basée sur la capacité mutli-services de ces objets, beaucoup plus complexe. En outre, ces applications ont des exigences de QoS devant être satisfaites ce qui fait du problème de la découverte et de la sélection des services IoT appropriés un problème de décision NP-difficile.

L'objectif de ce travail de thèse était de proposer et d'implémenter de nouveaux mécanismes et méthodes pour la mise en œuvre de nouveaux types d'applications et de nouveaux services IoT à valeur ajoutée. Notre objectif principal était de concevoir une architecture IoT distribuée de type fog/cloud computing qui permette de localiser et de sélectionner les services fournis par les objets connectés hétérogènes et qui permettent de satisfaire un objectif complexe, en prenant comme domaine d'application celui des véhicules connectés.

Afin de répondre à cet objectif, nous avons tout d'abord effectué un état de l'art sur les volets essentiels de notre travail : les notions préliminaires autour du domaine de l'IoT, du WoT, des services Web sémantiques et du fog/cloud computing et des méthodes de découverte, de sélection et de composition de services. Cette phase a été des plus importantes. Elle nous a permis de nous faire une idée globale sur notre sujet, d'identifier les terminologies et de déterminer les éléments nécessaires pour concevoir nos solutions.

Le diagramme de la Figure 5.34 donne un aperçu générique sur nos contributions de la virtualisation des objets IoT hétérogènes sur le web à la découverte et la sélection des objets qui permettent de réaliser un objectif donné.

Dans la première étape, nous avons exprimé notre première contribution qui consiste en la conception d'une architecture IoT distribuée de haut niveau basée sur des artefacts logiciels appelés *avatars autonomes*. Nous nous sommes concentrés sur la conception d'une ontologie modulaire qui décrit l'avatar dans ses divers aspects et d'un module de raisonnement basé sur des règles logiques. L'autonomie de l'avatar lui permet de découvrir et de sélectionner les avatars qui peuvent collaborer avec lui pour accomplir ses objectifs complexes.

Pour la méthode de découverte, i.e., notre deuxième contribution, nous avons proposé une nouvelle approche distribuée qui repose sur des mécanismes de réseautage social et les algorithmes de clustering fuzzy c-means. Ceci permet une découverte

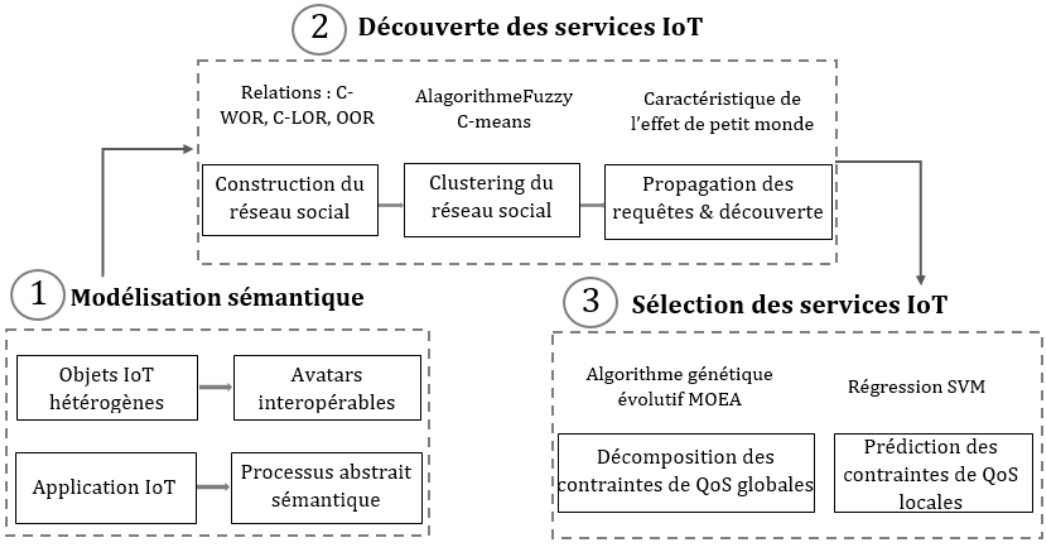


FIGURE 5.34 – Aperçu générique de nos contributions

efficace de services fournis par des milliards d'objets afin de mener des applications complexes d'une manière collaborative. Le réseautage social permet de regrouper les avatars les plus à même de coordonner et de réaliser l'application IoT demandée en se basant sur des mesures sociales comme l'emplacement et les intérêts communs. Un algorithme de clustering permet alors de classer les avatars sociaux en plusieurs clusters en fonction de leurs fonctionnalités pour guider au mieux l'acheminement des demandes de découverte vers les avatars pouvant satisfaire les requêtes envoyées.

En ce qui concerne la sélection des services découverts dans la phase précédente, deux contributions sont proposées. La première consiste en une méthode basée sur la décomposition des contraintes de QoS globales en contraintes locales en tenant compte de la fluctuation de QoS des avatars candidats. Elle est mise en œuvre via un algorithme génétique évolutionnaire multi-objectif (MOEA). Quant à la deuxième proposition, elle est basée sur la prédiction des contraintes de QoS locales en utilisant la méthode de régression SVM. Les contraintes locales obtenues, par l'algorithme MOEA ou la méthode SVM, sont utilisées comme bornes supérieures/inférieures pour effectuer une sélection locale distribuée sur les différents clusters d'avatars.

Afin de prouver l'efficacité et la scalabilité de la solution proposée, des expérimentations non-fonctionnelles et un cas d'utilisation autour du scénario de dépassement des véhicules connectés ont été mis en place. Nos propositions ont été mises en œuvre via une architecture microservices pour fournir un cadre générique et léger d'interopérabi-

lité et de modularité. En se basent sur cette architecture, plusieurs études d'évaluation de performances ont été menées en considérant un système à grande échelle d'avatars. Les résultats obtenus sont encourageants et montrent que nos approches de découverte et de sélection se comportent efficacement vis-à-vis d'autres méthodes.

Perspectives

Bien que diverses fonctionnalités aient été accomplies dans ce travail, plusieurs perspectives à court et à long terme peuvent être citées comme une poursuite de ce travail de thèse.

À court terme

1. **Expression des objectifs d'avatars :** dans notre approche proposée, les objectifs d'avatars (applications IoT) sont modélisés via des processus métiers sémantiques. Cependant, cette modélisation est effectuée par le concepteur ce qui rend cette tâche lourde et fastidieuse. Par conséquent, il serait intéressant d'explorer des mécanismes et des techniques qui permettent de traduire des besoins d'avatars exprimés sous une forme logique simple ou en langage naturel en un processus métier. Parmi les techniques proposées en littératures, nous retrouvons les travaux de [Zhovtobryukh, 2007] et de [Pradel et al., 2014].
2. **Confiance sociale :** comme les systèmes IoT sont ouverts, chaque avatar peut publier des services ou envoyer des requêtes pour solliciter et utiliser les services fournis par leurs homologues. Cependant, le fait de ne pas avoir d'information sur la fiabilité d'un avatar fournisseur de services n'empêche pas l'avatar consommateur de pouvoir effectuer un mauvais choix et choisir un avatar malhonnête qui fournit des services de mauvaise qualité, fastidieux, coûteux, voire nuisibles. Par conséquent, des mécanismes de confiance et de réputation doivent être intégrés lors du calcul des réseaux sociaux. Nous proposons d'utiliser la confiance comme une métrique sociale à considérer en plus des métriques basées sur : les intérêts (C-WOR), la location (C-LOC) et l'appartenance (OOR) dans notre modèle social. Cette métrique pourrait être calculée en se basant, par exemple, sur trois paramètres de confiance, à savoir : l'honnêteté, la coopération et les intérêts de la communauté.
3. **Gestion de la mobilité :** le domaine de l'IoT est caractérisé par sa nature incertaine où les objets peuvent apparaître et disparaître d'une manière dynamique. Dans ce travail, la mobilité n'est pas gérée parce qu'on suppose que les tâches de découverte et de sélection sont effectuées d'une manière réactive dans des laps de temps réduits. Cependant, dans un contexte hautement mobile comme

le contexte de l'IoT, il est très important de considérer cette caractéristique notamment dans la gestion du déploiement des avatars dans les nœuds fog et cloud et leur migration d'un nœud à un autre avec la mobilité.

À long terme

1. **Composition dynamique :** dans ce travail de thèse, nous nous sommes principalement concentrés sur les étapes de découverte et de sélection des services IoT. En ce qui concerne la composition, nous nous sommes basés sur un plan d'exécution construit via la collaboration entre les avatars. Cependant, nous n'avons pas considéré la supervision et l'adaptation de l'exécution du plan de composition construit.

   Ceci permettrait de modifier dynamiquement le processus de composition sans interrompre l'exécution. Plusieurs problématiques doivent être considérées durant cette étape :

   - Comment filtrer les avatars afin d'en obtenir une liste éligible ?
   - Quand remplacer un avatar dans la composition ?
   - Comment sélectionner l'avatar le plus adéquat parmi la liste précédemment obtenue durant les phases de découverte et de sélection ?
   - Comment remplacer un avatar indisponible ?
   - Comment considérer l'impact de la mobilité dans l'adaptation ?

2. **Déploiement dans une infrastructure réelle :** l'évaluation actuelle de nos différentes contributions autour de la découverte et de la sélection des services IoT est réalisée sur un serveur pour déterminer leurs performances. Cette approche devrait être complétée par un déploiement dans une infrastructure IoT réelle composée de plusieurs nœuds fog et un serveur cloud pour avoir une évaluation plus précise et plus réaliste de nos contributions.

Annexe A

Modélisations et données de scénario de dépassement

A.1 Modélisation sémantique du processus de dépassement

```
<owl:NamedIndividual rdf:about="&AvatarOnt;Process">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Overtaking
</hasLabel>
<hasChildTask rdf:resource="&AvatarOnt;TaskG1"/>
<hasChildTask rdf:resource="&AvatarOnt;Task3"/>
<hasChildTask rdf:resource="&AvatarOnt;Task4"/>
<hasChildTask rdf:resource="&AvatarOnt;Task5"/>
<hasChildTask rdf:resource="&AvatarOnt;TaskG2"/>
<hasChildTask rdf:resource="&AvatarOnt;Task9"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Sequence"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;TaskG1">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasChildTask rdf:resource="&AvatarOnt;Task1"/>
<hasChildTask rdf:resource="&AvatarOnt;Task2"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Parallel"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;TaskG2">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasChildTask rdf:resource="&AvatarOnt;Task6"/>
<hasChildTask rdf:resource="&AvatarOnt;TaskG3"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Parallel"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;TaskG3">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasChildTask rdf:resource="&AvatarOnt;Task7"/>
<hasChildTask rdf:resource="&AvatarOnt;Task8"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Sequence"/>
```

```
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task1">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label1</hasLabel>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task2">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label2</hasLabel>
 </owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task3">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label3</hasLabel>
<hasPrecondition rdf:datatype="&XMLSchema;string">Task1ANDTask2
</hasPrecondition>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task4">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label4</hasLabel>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task5">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label5</hasLabel>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task6">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label6</hasLabel>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task7">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label7</hasLabel>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="AvatarOnt;Task8">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label8</hasLabel>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task9">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label9</hasLabel>
 </owl:NamedIndividual>
```

Listing A.1 – Description sémantique du processus de dépassement

A.2 Modélisation sémantique de l'avatar A_1

```
<owl:NamedIndividual rdf:about="&AvatarOnt#Vehicle A">
<rdf:type rdf:resource="&AvatarOnt;Avatar"/>
<hasGoal rdf:resource="&AvatarOnt;Overtaking"/>
<hasInterest rdf:datatype="&XMLSchema;string">Transport/0.6</hasInterest>
<hasInterest rdf:datatype="&XMLSchema;string">Weather/0.2</hasInterest>
<hasInterest rdf:datatype="&XMLSchema;string">Obstacle Detection/0.6</hasInterest>
<hasOwner rdf:datatype="&XMLSchema;string">Driver1</hasOwner>
<hasLocation rdf:datatype="&XMLSchema;string">1.441185/43.57834
</hasLocation>
<represents rdf:resource="&AvatarOnt;Vehicle A"/>
<deployedIn rdf:datatype="&XMLSchema;anyURI">http://localhost:3001/
</deployedIn>
<hasService rdf:resource="&AvatarOnt;Service3"/>
<hasService rdf:resource="&AvatarOnt;Service4"/>
<hasService rdf:resource="&AvatarOnt;Service5"/>
<hasService rdf:resource="&AvatarOnt;Service8"/>
<hasService rdf:resource="&AvatarOnt;Service9"/>

</owl:NamedIndividual>
```

Listing A.2 – Description sémantique de l'avatar A_1

A.3 Modélisation sémantique du processus de dépassement après la phase de découverte

```
<owl:NamedIndividual rdf:about="&AvatarOnt;Process">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Overtaking</hasLabel>
<hasChildTask rdf:resource="&AvatarOnt;TaskG1"/>
<hasChildTask rdf:resource="&AvatarOnt;Task3"/>
<hasChildTask rdf:resource="&AvatarOnt;Task4"/>
<hasChildTask rdf:resource="&AvatarOnt;Task5"/>
<hasChildTask rdf:resource="&AvatarOnt;TaskG2"/>
<hasChildTask rdf:resource="&AvatarOnt;Task9"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Sequence"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;TaskG1">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasChildTask rdf:resource="&AvatarOnt;Task1"/>
<hasChildTask rdf:resource="&AvatarOnt;Task2"/>
```

```
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Parallel"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;TaskG2">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasChildTask rdf:resource="&AvatarOnt;Task6"/>
<hasChildTask rdf:resource="&AvatarOnt;TaskG3"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Parallel"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;TaskG3">
<rdf:type rdf:resource="&DEMISA;GroupedTask"/>
<hasChildTask rdf:resource="&AvatarOnt;Task7"/>
<hasChildTask rdf:resource="&AvatarOnt;Task8"/>
<DEMISA:hasGrouping rdf:resource="&AvatarOnt;Sequence"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task1">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label1</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar4</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar10</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar8</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar16</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar15</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar18</IsRealizedBy>

</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task2">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label2</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar2</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar10</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar8</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar16</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar15</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar13</IsRealizedBy>
 </owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task3">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label3</hasLabel>
<hasPrecondition rdf:datatype="&XMLSchema;string">Task1ANDTask2
</hasPrecondition>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar7</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar8</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar10</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar12</IsRealizedBy>
```

A.3 Modélisation sémantique du processus de dépassement après la phase de découverte

```
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar16</IsRealizedBy>
 </owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task4">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label4</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar2</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar7</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar10</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar13</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar17</IsRealizedBy>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task5">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label5</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar2</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar3</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar4</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar12</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar13</IsRealizedBy>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task6">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label6</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar8</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar10</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar7</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar16</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar17</IsRealizedBy>
 </owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task7">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label7</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar4</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar7</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar10</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar15</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar16</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar14</IsRealizedBy>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="AvatarOnt;Task8">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label8</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar2</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar3</IsRealizedBy>
```

```
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar4</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar12</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar13</IsRealizedBy>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&AvatarOnt;Task9">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/>
<hasLabel rdf:datatype="&XMLSchema;string">Label9</hasLabel>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar2</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar3</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar4</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar12</IsRealizedBy>
<IsRealizedBy rdf:datatype="&XMLSchema;string">Avatar13</IsRealizedBy>
 </owl:NamedIndividual>
```

Listing A.3 – Description sémantique du processus de dépassement après la découverte

A.4 Données QoS des avatars candidats

	L^1_{j1}		L^2_{j1}		L^3_{j1}	
Task	p^1_{j1}	f^1_{j1}	p^2_{j1}	f^2_{j1}	p^3_{j1}	f^3_{j1}
T_1	0.33	0.23	0.33	0.23	1	1
T_2	0.16	0.15	0.5	0.5	1	1
T_3	0.16	0.16	0.66	0.64	0.83	0.81
T_4	0.33	0.2	0.5	0.3	0.83	0.83
T_5	0.16	0.08	0.5	0.24	1	1
T_6	0.16	0.1	0.66	0.65	0.83	0.82
T_7	0.33	0.13	0.66	0.26	1	1
T_8	0.33	0.28	0.5	0.42	1	1
T_9	0.15	0.13	0.46	0.5	1	1

TABLE A.2 – Évaluation des niveaux de qualité candidats - Temps de réponse

A.4 Données QoS des avatars candidats

Tâche	Avatar	Temps de réponse	Moyenne	Débit	Moyenne
T_1	A_4	88, 7, 7, 51, 84, 71	51.33	8, 4, 21, 25, 25, 17	16.66
	A_{10}	61, 72, 15, 19, 93, 72	55.33	5, 7, 9, 25, 3, 15	10.66
	A_8	87, 7, 2, 59, 98, 58	61.83	26, 22, 27, 5, 21, 12	18.83
	A_{16}	31, 54, 98, 67, 20, 12	47	12, 2, 26, 13, 18, 26	16.16
	A_{15}	52, 61, 52, 35, 83, 48	55.16	1, 11, 20, 27, 5, 23	14.5
	A_{18}	79, 2, 64, 91, 18, 56	51.66	14, 15, 19, 2, 3, 12	10.83
T_2	A_2	44, 84, 69,85, 14, 81	62.83	15, 7, 18, 17, 2, 18	12.83
	A_{10}	30, 50, 16, 18, 78, 62	42.33	13, 1, 4, 28, 18, 5	11.5
	A_8	57, 47, 30, 92, 67, 95	64.66	19, 11, 20, 11, 10, 16	14.5
	A_{16}	75, 41, 62, 48,56, 37	53.16	18, 18, 24, 21, 22, 25	21.33
	A_{15}	19, 65, 37, 93, 42, 16	45.33	5, 10, 16, 15, 21, 5	12
	A_{13}	61, 1, 19, 78, 25,53	39.5	15, 22, 4, 17, 20, 6	14
T_3	A_1	84, 71, 43, 58, 86, 25	61.16	18, 13, 2, 22, 2, 10	11.16
	A_7	41, 15, 5, 37, 53, 25	29.33	27, 15, 9, 4, 25, 18	16.33
	A_8	43, 47, 8, 60, 26, 41	37.5	4, 8, 8, 19, 12, 16	11.16
	A_{10}	79, 20, 87, 58, 38, 98	63.33	1, 27, 9, 4, 8, 13	10.33
	A_{16}	34, 21, 68, 75, 70, 14	47	26, 2, 14, 13, 11, 10	12.66
	A_{12}	47, 49, 80, 82, 55, 38	58.5	15, 16, 12, 10, 18, 28	16.5
T_4	A_1	35, 29, 72, 26, 53, 68	47.16	14, 13, 20, 17, 8, 25	16.16
	A_2	86, 91, 96, 91 29, 17	68.33	26, 16, 2, 15, 27, 28	19
	A_7	48, 25, 23, 34, 72, 45	41.16	10, 21, 19, 24, 27, 18	19.83
	A_{10}	61, 23, 62, 77, 17, 45	47.5	6, 6, 27, 7, 15, 22	13.83
	A_{13}	35, 35, 51, 71, 65, 15	45.33	7, 5, 4, 21, 15, 6	9.66
	A_{17}	65, 64, 41, 45, 45, 20	46.66	22, 1, 4, 23, 12, 18	13.33

	L_{j1}^1		L_{j1}^2		L_{j1}^3	
Task	p_{j1}^1	f_{j1}^1	p_{j1}^2	f_{j1}^2	p_{j1}^3	f_{j1}^3
T_1	1	1	0.66	0.47	0.13	0.11
T_2	0.83	0.79	0.66	0.63	0.16	0.05
T_3	1	1	0.5	0.48	0.08	0.07
T_4	1	1	0.5	0.36	0.33	0.24
T_5	0.83	0.41	0.66	0.33	0.16	0.08
T_6	1	1	0.58	0.65	0.12	0.09
T_7	1	1	0.5	0.19	0.16	0.06
T_8	0.83	0.83	0.66	0.66	0.33	0.28
T_9	0.83	0.83	0.5	0.32	0.33	0.21

TABLE A.3 – Évaluation des niveaux de qualité candidats - Débit

Tâche	Avatar	Temps de réponse	Moyenne	Débit	Moyenne
T_5	A_1	8, 85, 69, 7, 21, 72	43.66	21, 3, 11, 24, 11, 26	16
	A_2	89, 32, 93, 4, 49, 38	50.83	23, 8, 3, 3, 2, 23	10.33
	A_3	67, 28, 91, 60, 32, 4	47	6, 22, 5, 13, 9, 11	11
	A_4	6, 80, 70, 60, 8, 77	50.16	12,24, 3, 9, 17, 28	15.5
	A_{12}	78, 93, 27, 46, 46, 70	60	6, 24, 14, 22, 2, 18	14.33
	A_{13}	37, 2, 68, 74, 69, 64	52.33	12, 27, 23, 9, 2, 15	14.66
T_6	A_8	74, 64, 12, 50, 92, 11	50.5	11, 9, 19, 24, 26, 22	18.5
	A_{10}	87, 34, 49, 32, 5, 53	43.33	21, 1, 22, 13, 25, 12	15.66
	A_7	22, 52, 31, 56, 2, 55	36.33	3, 26, 1, 15, 25, 7	12.83
	A_{16}	94, 71, 71, 12, 20, 32	50	14, 13, 26, 11, 21, 4	14.83
	A_{17}	19, 11, 32, 88, 90, 90	55	6, 6, 16, 18, 3, 11	10
	A_{15}	68, 74, 15, 90, 12, 20	46.5	15, 23, 10, 25, 17, 16	17.66
T_7	A_4	33, 6, 40, 22, 28, 20	24.83	16, 3, 28, 5, 16, 19	14.5
	A_7	80, 8, 88, 72, 20, 98	61	25, 16, 3, 5, 25, 23	16.16
	A_{10}	32, 84, 49, 24, 13, 97	49.83	2, 7, 24, 8, 8, 11	10
	A_{15}	12, 78, 15, 30, 99, 63	49.5	3, 18, 26, 19, 23, 21	18.33
	A_{16}	66, 15, 15, 17, 94, 73	46.66	5, 16, 27, 23,21, 22	19
	A_{14}	66, 32, 32, 91 ,27, 45	48.83	8, 8, 15, 15, 4, 16	11
T_8	A_1	63,2, 17, 44, 26, 50	33.66	3, 5, 18, 17, 8, 20	11.83
	A_2	86, 97, 12, 47, 91, 68	66.83	27, 26, 13, 13, 8, 13	16.66
	A_3	57, 75, 77, 55, 31, 24	53.16	11, 24, 6, 25, 19, 25	18.33
	A_4	78, 80, 96, 8, 72, 22	59.33	7, 17, 5, 24, 17, 2	12
	A_{12}	51, 95, 73, 72, 46, 91	71.33	6, 21, 9, 12, 17, 22	14.5
	A_{13}	35, 17, 53, 88, 79, 41	52.16	3, 14, 1, 9, 7, 6	6.66
T_9	A_1	73, 18, 29, 37, 10,77	40.66	7, 17, 27, 14, 24, 20	18.16
	A_2	53, 43, 75, 9, 24, 65	44.83	24, 6, 22, 10, 21, 25	18
	A_3	24, 50, 23, 13, 12, 46	28	7, 24, 8, 21, 6, 21	14.5
	A_4	42, 29, 46, 36, 65, 2	36.66	5, 20, 12, 22, 4, 26	14.83
	A_{12}	44, 47, 87, 43, 7, 73	50.16	16, 29, 23, 9, 25, 27	21.5
	A_{13}	73, 37, 91, 93, 83, 41	69.66	10, 20, 19, 21, 17, 5	15.33

TABLE A.1 – Propriétés QoS des avatars candidats pour chaque tâche

A.4 Données QoS des avatars candidats

Cas	Localisation	Heure	Date	$dist_1(C_0, C_i)$	$dist_2(C_0, C_i)$	SIM(C_0, C_i)
1	42.765, 0.377	11H	01/01/2020	0.010	0.125	0.056
2	42.865, 0.773	14H	05/01/2020	0.015	0.0	0.009
3	43.236, 1.021	13H	08/01/2020	0.054	0.041	0.049
4	43.976, 1.196	13H	10/01/2020	0.014	0.041	0.025
5	43.725, 1.726	15H	01/02/2020	0.016	0.041	0.026
6	42.740, 1.047	19H	11/02/2020	0.014	0.208	0.092
7	42.937, 1.872	22H	13/02/2020	0.013	0.333	0.141
8	44.154, 1.230	14H	15/02/2020	0.010	0.0	0.006
9	43.373, 2.059	13H	20/02/2020	0.013	0.041	0.024
10	43.611, 0.420	20H	25/02/2020	0.014	0.250	0.108
11	43.483, 2.353	17H	11/03/2020	0.010	0.125	0.056
12	43.667, 2.105	11H	21/03/2020	0.011	0.125	0.057
13	44.215, 1.368	14H	24/03/2020	0.010	0.0	0.006
14	42.952, 0.752	10H	02/04/2020	0.017	0.166	0.077
15	42.991, 0.193	14H	12/04/2020	0.011	0.0	0.006
16	44.204, 1.118	09H	14/04/2020	0.010	0.208	0.089
17	42.553, 1.309	18H	16/04/2020	0.011	0.166	0.073
18	43.955, 1.104	17H	22/04/2020	0.014	0.125	0.058
19	43.269, 1.998	20H	01/05/2020	0.014	0.250	0.108
20	42.971, 1.813	15H	11/05/2020	0.014	0.041	0.025
21	44.016, 0.819	17H	13/05/2020	0.012	0.125	0.057
22	42.619, 0.561	11H	14/05/2020	0.010	0.125	0.056
23	43.504, 1.614	16H	19/05/2020	0.024	0.083	0.048
24	43.427, 1.013	10H	18/06/2020	0.059	0.166	0.102
25	43.195, 1.001	14H	20/06/2020	0.043	0.0	0.026
26	42.998, 1.444	18H	22/06/2020	0.022	0.166	0.079
27	43.436, 1.801	10H	29/06/2020	0.019	0.166	0.078
28	43.261, 1.300	14H	01/07/2020	0.067	0.0	0.040
29	43.802, 0.183	13H	11/07/2020	0.010	0.041	0.022
30	42.926, 2.203	18H	21/07/2020	0.010	0.166	0.072

TABLE A.4 – Caractéristiques et distances des cas historiques par rapport au cas cible

Tâche	Cas Temps de réponse	Cas Débit
T_1	62.68, 61.15, 62.29, 58.75, 60.97, 59.56, 57.47, 59.02, 60.45, 57.78, 57.28, 57.78, 61.79, 58.67, 61.88, 60.66, 61.24, 62.53, 59.75, 62.62.	18.01, 17.74, 17.42, 16.77, 16.26, 16.84, 18.27, 18.11, 18.38, 16.77, 17.13, 16.03, 16.38, 17.85, 16.72, 17.67, 16.58, 16.96, 16.4, 16.96.
T_2	43.03, 43.88, 46.24, 42.08, 46.91, 44.69, 43.04, 45.6, 43.5, 43.61, 46.45, 46.24, 46.4, 45.3, 44.86, 46.81, 46.75, 45.59, 42.69, 45.96.	11.15, 12.0, 12.84, 12.45, 11.16, 12.84, 11.03, 11.04, 12.62, 11.68, 12.94, 12.62, 12.66, 11.21, 11.22, 12.11, 11.6, 11.08, 11.62, 11.04.
T_3	56.71, 59.51, 58.36, 59.78, 56.94, 57.51, 56.03, 59.81, 58.08, 58.09, 58.3, 58.88, 57.45, 56.77, 57.18, 56.81, 57.9, 58.67, 56.45, 58.92.	15.41, 14.63, 15.4, 17.87, 16.63, 14.89, 15.57, 16.59, 16.69, 16.34, 17.12, 14.12, 15.94, 17.82, 15.94, 14.3, 15.68, 17.5, 17.45, 16.34.
T_4	46.69, 45.66, 47.73, 47.06, 44.13, 45.07, 47.93, 44.78, 45.56, 47.33, 45.74, 44.45, 44.63, 45.02, 46.99, 46.32, 47.67, 45.23, 46.58, 46.55.	17.45, 17.95, 14.71, 16.01, 15.02, 15.7, 17.81, 17.59, 15.68, 15.61, 15.28, 17.0, 16.01, 16.6, 15.4, 14.95, 15.89, 14.85, 14.98, 15.84.
T_5	45.22, 42.69, 42.38, 44.19, 45.37, 45.65, 44.0, 42.14, 45.53, 45.6, 44.34, 42.55, 43.01, 44.62, 43.42, 44.29, 45.42, 43.5, 45.34, 45.37	17.43, 17.3, 16.23, 15.13, 15.87, 16.4, 16.15, 16.26, 16.44, 16.32, 16.7, 17.36, 16.32, 14.84, 14.59, 15.41, 17.27, 15.14, 17.94, 16.58
T_6	48.34, 50.21, 50.36, 50.11, 49.85, 48.85, 51.06, 48.27, 51.8, 51.63, 49.66, 51.55, 49.23, 48.0, 51.82, 50.65, 48.05, 50.25, 52.0, 51.19.	14.14, 15.08, 14.57, 14.47, 15.55, 15.99, 15.46, 15.38, 15.86, 14.58, 15.01, 15.69, 15.35, 15.42, 14.65, 15.35, 15.24, 15.73, 14.98, 15.84.
T_7	46.71, 48.44, 45.25, 46.56, 48.14, 48.63, 48.51, 46.26, 47.54, 46.06, 47.28, 48.69, 46.89, 47.83, 48.78, 48.9, 46.21, 46.94, 48.46, 46.69	14.74, 16.68, 15.25, 14.92, 14.43, 16.05, 15.18, 14.6, 14.49, 15.91, 15.09, 15.09, 16.87, 15.54, 14.01, 16.74, 14.49, 15.13, 14.39, 15.45
T_8	52.42, 52.92, 51.54, 53.96, 53.86, 51.11, 52.04, 52.06, 52.49, 51.25, 53.5, 52.27, 53.93, 53.12, 51.18, 53.59, 53.74, 51.63, 52.69, 51.19	15.47, 17.81, 16.47, 17.06, 15.75, 17.28, 16.4, 16.38, 17.6, 17.95, 16.29, 17.94, 17.4, 15.03, 16.57, 16.79, 16.55, 17.15, 16.31, 17.88
T_9	42.95, 41.86, 44.81, 40.19, 42.46, 42.1, 43.56, 43.33, 43.59, 44.86, 43.77, 41.55, 41.88, 43.04, 41.54, 41.99, 40.99, 43.8, 40.6, 40.31	18.49, 18.8, 18.44, 17.0, 18.24, 18.22, 18.31, 18.04, 18.82, 17.78, 18.09, 17.56, 17.01, 18.32, 18.77, 18.88, 18.21, 17.35, 17.45, 17.37

TABLE A.5 – Contraintes locales historiques similaires au cas cible

Annexe B

Indices d'évaluation

B.1 Indices d'évaluation de clustering

B.1.1 L'indice Xie and Beni (XB)

L'indice Xie and Beni (XB) [Xie and Beni, 1991] représente un critère de validation qui permet de mesurer la qualité des méthodes de clustering flou comme l'algorithme Fuzzy C-means. Il est définit comme la distance carrée minimale qui mesure la séparation inter-cluster (dissemblance entre les clusters résultants) et la distance quadratique moyenne entre chaque objet de données et son centre de cluster qui mesure la compacité intra-cluster (ressemblance entre les individus du même cluster). Le nombre optimal de clusters à prendre est lorsque cet indice prend la valeur minimale. Il est à noter que plus que la valeur XB est petit, plus que les résultats de clustering sont meilleurs. Il est calculé comme indiqué en Équation B.1.

$$XB = \frac{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 d_{ij}^2}{n.min_{j=/=i} abs(v_j - v_i)^2} \tag{B.1}$$

B.1.2 L'indice Partition Coefficient (PC)

L'indice Partition Coefficient (PC) [Bezdek, 1974] [Wang and Zhang, 2007] est un indice de validité utilisé pour mesurer le flou de la partition des clusters finaux obtenus à la fin de l'algorithme de clustering. Contrairement à l'indice XB, cet indice n'intègre pas la mesure intra-clusters nécessaire à l'évaluation de la compacité des clusters obtenus. Le nombre optimal de clusters à considérer est tel que cet indice prend une valeur maximale. Il est à noter que plus la valeur PC est élevée, meilleurs sont les résultats de la partition. Il est défini comme la norme de Frobenius (norme euclidienne ou hermitienne standard) de la matrice membership, divisée par le nombre de points de données considérées. L'indice PC est calculé comme indiqué en Équation B.2.

$$PC = \frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 \tag{B.2}$$

B.1.3 L'indice Partition Entropy (PE)

L'indice Partition Entropy (PE) [Tran and Wagner, 2000] [Wang and Zhang, 2007] représente un indice qui mesure l'entropie, i.e., le degré de désorganisation et d'imprédictibilité des résultats d'un algorithme de clustering. Il s'agit d'une mesure scalaire de la quantité de flou dans la matrice de memership obtenue dans la derrière phase de partition. Plus que la valeur de cet indice PE est minimale, meilleurs sont les résultats de clustering. PE est calculé comme indiqué en Équation B.3

$$PE = \frac{-1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 \log_2(u_{ij}) \tag{B.3}$$

B.2 Indices d'évaluation de MOEA

B.2.1 Inverted Generational Distance (IGD)

L'indice de la distance générationnelle inversée (IGD) [Fan et al., 2017] [Bezerra et al., 2017] est une métrique qui permet d'évaluer les performances des algorithmes d'optimisation évolutive multi-objectifs (MOEA). La mesure IGD représente la distance moyenne de chaque point de référence à la solution la plus proche dans l'ensemble de solutions. Autrement dit, il s'agit de la distance approximative du front de Pareto à l'ensemble de solutions. Cet indice est calculé comme indiqué en Équation B.4.

$$lIGD\,(P^*, A) = \frac{\sum_{y^* \in P^*} d\,(y^*, A)}{|P^*|} \tag{B.4}$$

Tel que :

$$d\,(y^*, A) = \min_{y \in A} \{ \sqrt{\sum_{i=1}^{m} (y_i^* - y_i)^2} \} \tag{B.5}$$

Où P^* est un ensemble de points de référence dans la population optimale, A est une population intermédiaire. La plus petite valeur de IGD indique la meilleure performance de la convergences et de la diversité.

B.2.2 Hyper-volume (HV)

L'indice d'hyper-volume (HV) [Wagner and Trautmann, 2010], également appelée Lebesgue, est une métrique utilisée pour mesurer à la fois la proximité des solutions obtenues par l'algorithme évolutive de l'ensemble des solutions Pareto et la diversité des solutions obtenues. Plus précisément, l'indice HV calcule le volume de l'espace objectif couvert par les membres de l'ensemble des solutions Pareto obtenues. L'hyper-volume (HV) est mathématiquement calculé comme indiqué en Équation B.6.

$$\begin{cases} I_H^- (A, P^*, R) = I_H (P^*, R) - I_H(A, R) \\ I_H (P^*, R) = Vol_{v \in P^*}(v) \\ I_H(A, R) = Vol_{v \in A}(v) \end{cases} \tag{B.6}$$

La plus petite valeur de (I_H^-) indique la meilleure performance de la convergences et de la diversité.

B.3 Indices d'évaluation de régression SVM

B.3.1 Mean squared error (MSE)

L'erreur quadratique moyenne (MSE) [Prasad and Rao, 1990] est définit comme une fonction positive qui permet de mesurer la moyenne des carrés des erreurs d'un processus de prédiction et d'estimation. Autrement dit, cette métrique représente la différence quadratique moyenne entre les valeurs estimées et les valeurs réelles obtenues. Elle est calculée comme indiqué en Équation B.7.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (X_i - Y_i)^2 \tag{B.7}$$

Tel que : X est le vecteur des valeurs observées de la variable prédite et Y le vecteur des valeurs prédites.

B.3.2 Coefficient de corrélation quadratique ρ^2

Le coefficient de corrélation quadratique, noté ρ^2, [Asuero et al., 2006] est une métrique qui permet de mesurer la corrélation linéaire entre deux variables quantitatives. Mathématiquement, Le coefficient de corrélation entre 2 variables quantitatives X et Y est égal au rapport de la covariance entre X et Y divisé par le produit des écart-types de X et Y. Le coefficient ρ est calculé comme indiqué en Équation B.8.

$$\rho = \frac{cov(X, Y)}{\sqrt{var(X) - var(Y)}} \tag{B.8}$$

Bibliographie

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based reasoning : Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1) :39–59. (Cité en page 113.)

[Abras, 2009] Abras, S. (2009). Système domotique multi-agents pour la gestion de l'énergie dans l'habitat. *These de l'institut polytechnique de Grenoble*. (Cité en page 28.)

[Afshari et al., 2010] Afshari, A., Mojahed, M., and Yusuff, R. M. (2010). Simple additive weighting approach to personnel selection problem. *International Journal of Innovation, Management and Technology*, 1(5) :511. (Cité en page 92.)

[Akkiraju et al., 2005] Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., and Verma, K. (2005). Web service semantics-wsdl-s. (Cité en page 18.)

[Alaya, 2014] Alaya, M. B. (2014). *Towards interoperability, self-management, and scalability for machine-to-machine systems*. PhD thesis, PhD thesis, INSA de Toulouse, 2014.(Cité efinpage 20.). (Cité en page 3.)

[Alaya et al., 2015] Alaya, M. B., Medjiah, S., Monteil, T., and Drira, K. (2015). Towards semantic data interoperability in onem2m standard. (Cité en page 51.)

[Ali et al., 2018] Ali, S., Kibria, M. G., Jarwar, M. A., Lee, H. K., and Chong, I. (2018). A model of socially connected web objects for iot applications. *Wireless Communications and Mobile Computing*, 2018. (Cité en page 70.)

[Allweyer, 2016] Allweyer, T. (2016). *BPMN 2.0 : introduction to the standard for business process modeling*. BoD–Books on Demand. (Cité en pages 38 et 42.)

[Alrifai et al., 2012] Alrifai, M., Risse, T., and Nejdl, W. (2012). A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2) :1–31. (Cité en pages 37, 42 et 95.)

[Alrifai et al., 2010] Alrifai, M., Skoutas, D., and Risse, T. (2010). Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20. ACM. (Cité en page 36.)

[Amato et al., 2016] Amato, Flora, M., and Francesco (2016). Pattern-based orchestration and automatic verification of composite cloud services. *Computers & Electrical Engineering*, 56 :842–853. (Cité en page 38.)

[Analytics, 2018] Analytics, I. (2018). State of the iot 2018 : Number of iot devices now at 7b – market accelerating. disponible sur : https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/. (Cité en page 1.)

[Ankolekar et al., 2001] Ankolekar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D. L., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., et al. (2001). Daml-s : Semantic markup for web services. (Cité en page 18.)

[Arora et al., 2019] Arora, J., Khatter, K., and Tushir, M. (2019). Fuzzy c-Means Clustering Strategies : A Review of Distance Measures. In Hoda, M. N., Chauhan, N., Quadri, S. M. K., and Srivastava, P. R., editors, *Software Engineering*, volume 731, pages 153–162. Springer Singapore, Singapore. (Cité en page 131.)

[Arpinar et al., 2005] Arpinar, I. B., Zhang, R., Aleman-Meza, B., and Maduko, A. (2005). Ontology-driven web services composition platform. *Information Systems and E-Business Management*, 3(2) :175–199. (Cité en pages 40 et 42.)

[Asuero et al., 2006] Asuero, A. G., Sayago, A., and Gonzalez, A. (2006). The correlation coefficient : An overview. *Critical reviews in analytical chemistry*, 36(1) :41–59. (Cité en page 173.)

[Atzori et al., 2011] Atzori, L., Iera, A., and Morabito, G. (2011). Siot : Giving a social structure to the internet of things. *IEEE communications letters*, 15(11) :1193–1195. (Cité en pages 72 et 75.)

[Atzori et al., 2014] Atzori, L., Iera, A., and Morabito, G. (2014). From" smart objects" to" social objects" : The next evolutionary step of the internet of things. *IEEE Communications Magazine*, 52(1) :97–105. (Cité en page 69.)

[Atzori et al., 2012] Atzori, L., Iera, A., Morabito, G., and Nitti, M. (2012). The social internet of things (siot)–when social networks meet the internet of things : Concept, architecture and network characterization. *Computer networks*, 56(16) :3594–3608. (Cité en page 68.)

[Aversano et al., 2004] Aversano, L., Canfora, G., and Ciampi, A. (2004). An algorithm for web service discovery through their composition. In *Proceedings. IEEE International Conference on Web Services, 2004.*, pages 332–339. IEEE. (Cité en pages 40 et 42.)

[Aïssaoui, 2018] Aïssaoui, F. (2018). *Autonomic Approach based on Semantics and Checkpointing for IoT System Management*. PhD thesis, Capitole Toulouse. Thèse de doctorat dirigée par Monteil, Thierry et Tazi, Saïd Informatique Toulouse 1 2018. (Cité en page 3.)

[Bach, 2006] Bach, T. L. (2006). *Construction d'un Web sémantique multi-points de vue*. PhD thesis, Paris, ENMP. (Cité en page 21.)

[Barnes, 1954] Barnes, J. A. (1954). Class and committees in a norwegian island parish. *Human relations*, 7(1) :39–58. (Cité en page 68.)

[Batra and Bawa, 2011] Batra, S. and Bawa, S. (2011). Semantic discovery of web services using principal component analysis. *Physical Sciences*, 6(18) :4466–4472. (Cité en page 32.)

[Benghozi et al., 2008] Benghozi, P.-J., Bureau, S., and Massit-Folea, F. (2008). L'internet des objets. quels enjeux pour les européens ? (Cité en page 12.)

[Bennett et al., 2016] Bennett, J. D., Hemani, M. S., and O'farrell, W. G. (2016). Business process execution language program simulation. US Patent 9,256,516. (Cité en pages 38 et 42.)

[Bensaid et al., 1997] Bensaid, N., Mathieu, P., et al. (1997). A framework for cooperation in hierarchical multi-agent systems. *Mathematical Modeling and Scientific Computing*, 8. (Cité en page 27.)

[Bergmann, 2003] Bergmann, R. (2003). *Experience management : foundations, development methodology, and internet-based applications*, volume 2432. Springer. (Cité en page 114.)

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific american*, 284(5) :28–37. (Cité en page 20.)

[Berrani et al., 2018] Berrani, S., Yachir, A., Djamaa, B., and Aissani, M. (2018). Multi-agent system based service composition in the internet of things. In Amine, A., Mouhoub, M., Ait Mohamed, O., and Djebbar, B., editors, *Computational Intelligence and Its Applications*, pages 521–532, Cham. Springer International Publishing. (Cité en pages 35 et 42.)

[Bezdek, 1974] Bezdek, J. C. (1974). Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology*, 1(1) :57–71. (Cité en page 171.)

[Bezdek, 1981] Bezdek, J. C. (1981). Objective function clustering. In *Pattern recognition with fuzzy objective function algorithms*, pages 43–93. Springer. (Cité en page 82.)

[Bezerra et al., 2017] Bezerra, L. C., López-Ibáñez, M., and Stützle, T. (2017). An empirical assessment of the properties of inverted generational distance on multi-and many-objective optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 31–45. Springer. (Cité en page 172.)

[Bizer et al., 2011] Bizer, C., Heath, T., and Berners-Lee, T. (2011). Linked data : The story so far. In *Semantic services, interoperability and web applications : emerging concepts*, pages 205–227. IGI Global. (Cité en page 20.)

[Boley et al., 2010] Boley, H., Paschke, A., and Shafiq, O. (2010). Ruleml 1.0 : the overarching specification of web rules. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 162–178. Springer. (Cité en page 24.)

[Bonino et al., 2014] Bonino, D., Corno, F., and De Russis, L. (2014). Poweront : An ontology-based approach for power consumption estimation in smart homes. In *International Internet of Things Summit*, pages 3–8. Springer. (Cité en page 22.)

[Bonomi et al., 2012] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition*

of the MCC workshop on Mobile cloud computing, pages 13–16. ACM. (Cité en page 29.)

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. (Cité en page 118.)

[Cao et al., 2001] Cao, J., Kerbyson, D. J., and Nudd, G. R. (2001). High performance service discovery in large-scale multi-agent and mobile-agent systems. *International Journal of Software Engineering and Knowledge Engineering*, 11(05) :621–641. (Cité en page 33.)

[Chaib-Draa et al., 2001] Chaib-Draa, B., Jarras, I., and Moulin, B. (2001). Systèmes multi-agents : principes généraux et applications. *Edition Hermès*, 242 :1030–1044. (Cité en page 27.)

[Chan and Lyu, 2008] Chan, P. P. W. and Lyu, M. R. (2008). Dynamic web service composition : A new approach in building reliable web service. In *22nd International Conference on Advanced Information Networking and Applications (aina 2008)*, pages 20–25. IEEE. (Cité en pages 40 et 42.)

[Chareton et al., 2017] Chareton, C., Brunel, J., and Chemouil, D. (2017). Sur l'assignation de buts comportementaux à des coalitions d'agents. In *Approches Formelles dans l'Assistance au Développement de Logiciels*. (Cité en pages 34 et 42.)

[Charif, 2007] Charif, Y. (2007). *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs*. PhD thesis, Paris 6. (Cité en page 38.)

[Charlet et al., 2004] Charlet, J., Bachimont, B., and Troncy, R. (2004). Ontologies pour le web sémantique. *Revue I3, numéro Hors Série «Web sémantique*, pages 43–63. (Cité en page 21.)

[Chen et al., 2013] Chen, W., Paik, I., and Hung, P. C. (2013). Constructing a global social service network for better quality of web service discovery. *IEEE transactions on services computing*, 8(2) :284–298. (Cité en pages 72 et 75.)

[Cheng et al., 2002] Cheng, Z., Singh, M. P., and Vouk, M. A. (2002). Composition constraints for semantic web services. In *WWW2002 Workshop on Real World RDF and Semantic Web Applications*. (Cité en pages 39 et 42.)

[Chhun et al., 2016] Chhun, S., Moalla, N., and Ouzrout, Y. (2016). Qos ontology for service selection and reuse. *Journal of Intelligent Manufacturing*, 27(1) :187–199. (Cité en page 23.)

[Chiandussi et al., 2012] Chiandussi, G., Codegone, M., Ferrero, S., and Varesio, F. (2012). Comparison of multi-objective optimization methodologies for engineering applications. *Computers Mathematics with Applications*, 63(5) :912 – 942. (Cité en page 98.)

[Chifu et al., 2017] Chifu, V. R., Pop, C. B., Salomie, I., and Chifu, E. S. (2017). Hybrid honey bees mating optimization algorithm for identifying the near-optimal solution in web service composition. *Computing and Informatics*, 36(5) :1143–1172. (Cité en pages 36 et 42.)

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1. (Cité en page 18.)

[Christophe et al., 2011] Christophe, B., Boussard, M., Lu, M., Pastor, A., and Toubiana, V. (2011). The web of things vision : Things as a service and interaction patterns. *Bell labs technical journal*, 16(1) :55–61. (Cité en page 53.)

[Chun et al., 2005] Chun, S. A., Atluri, V., and Adam, N. R. (2005). Using semantics for policy-based web service composition. *Distributed and Parallel Databases*, 18(1) :37–64. (Cité en page 39.)

[Cisco, 2014] Cisco (2014). Cisco delivers vision of fog computing to accelerate value from billions of connected devices. Accessed : 2019-07-27. (Cité en page 29.)

[Compton et al., 2012] Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., et al. (2012). The ssn ontology of the w3c semantic sensor network incubator group. *Web semantics : science, services and agents on the World Wide Web*, 17 :25–32. (Cité en page 22.)

[Cristianini et al., 2000] Cristianini, N., Shawe-Taylor, J., et al. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press. (Cité en page 118.)

[Daniele et al., 2016] Daniele, L., Solanki, M., den Hartog, F., and Roes, J. (2016). Interoperability for smart appliances in the iot world. In *International Semantic Web Conference*, pages 21–29. Springer. (Cité en page 23.)

[Daudé, 2005] Daudé, E. (2005). Systèmes multi-agents pour la simulation en géographie : vers une géographie artificielle. (Cité en page 28.)

[De et al., 2011] De, S., Barnaghi, P., Bauer, M., and Meissner, S. (2011). Service modelling for the internet of things. In *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 949–955. IEEE. (Cité en pages 49 et 52.)

[de Azevedo et al., 2010] de Azevedo, R. R., Dantas, E. R. G., Freitas, F., Rodrigues, C., de Almeida, M., Veras, W. C., and Santos, R. (2010). An autonomic ontology-based multiagent system for intrusion detection in computing environments. *International Journal for Infonomics (IJI)*, 3(1) :182–189. (Cité en page 23.)

[Deighton et al., 2011] Deighton, J., Fader, P., Haenlein, M., Kaplan, A. M., Libai, B., and Muller, E. (2011). Médias sociaux et entreprise, une route pleine de défis

commentaires invités. *Recherche et Applications en Marketing (French Edition)*, 26(3) :117–124. (Cité en page 68.)

[Djemai et al., 2019] Djemai, T., Stolf, P., Monteil, T., and Pierson, J.-M. (2019). A discrete particle swarm optimization approach for energy-efficient iot services placement over fog infrastructures. In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 32–40. IEEE. (Cité en page 65.)

[Djemai et al., 2020] Djemai, T., Stolf, P., Monteil, T., and Pierson, J.-M. (2020). Mobility support for energy and qos aware iot services placement in the fog. In *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–7. IEEE. (Cité en page 65.)

[Dolui and Datta, 2017] Dolui, K. and Datta, S. K. (2017). Comparison of edge computing implementations : Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE. (Cité en page 30.)

[Domingue et al., 2005] Domingue, J., Roman, D., and Stollberg, M. (2005). Web service modeling ontology (wsmo)-an ontology for semantic web services. (Cité en page 19.)

[Dong and Chen, 2016] Dong, B. and Chen, J. (2016). Architecting semantics-based publish/subscribe applications for the internet of things. In *2016 International Forum on Management, Education and Information Technology Application*. Atlantis Press. (Cité en pages 33 et 42.)

[Duce, 2008] Duce, H. (2008). Internet of things in 2020. (Cité en page 12.)

[Durfee, 2001] Durfee, E. H. (2001). Distributed problem solving and planning. In *ECCAI Advanced Course on Artificial Intelligence*, pages 118–149. Springer. (Cité en pages 39 et 42.)

[Ebrahimifard et al., 2016] Ebrahimifard, A., Amiri, M. J., Arani, M. K., and Parsa, S. (2016). Mapping bpmn 2.0 choreography to ws-cdl : a systematic method. *Journal of E-Technology*, 7(1) :01–23. (Cité en pages 38 et 42.)

[El Falou, 2010] El Falou, M. (2010). *Contributions à la composition dynamique de services fondée sur des techniques de planification et diagnostic multi-agents*. PhD thesis, Université de Caen. (Cité en page 38.)

[Ermolayev et al., 2003] Ermolayev, V., Keberle, N., and Plaksin, S. (2003). Towards agent-based rational service composition–racing approach. In *International Conference on Web Services*, pages 167–182. Springer. (Cité en page 39.)

[Erol et al., 1995] Erol, K., Hendler, J. A., Nau, D. S., and Tsuneto, R. (1995). A critical look at critics in htn planning. Technical report. (Cité en page 39.)

[Fallatah et al., 2014] Fallatah, H., Bentahar, J., and Asl, E. K. (2014). Social network-based framework for web services discovery. In *2014 International Conference on Future Internet of Things and Cloud*, pages 159–166. IEEE. (Cité en pages 71 et 75.)

[Fan et al., 2015] Fan, W.-J., Yang, S.-L., Perros, H., and Pei, J. (2015). A multi-dimensional trust-aware cloud service selection mechanism based on evidential reasoning approach. *International Journal of Automation and Computing*, 12(2) :208–219. (Cité en page 35.)

[Fan et al., 2017] Fan, Z., Yi Fang, Li, W., Jiewei Lu, Cai, X., and Caimin Wei (2017). A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 209–216. (Cité en pages 98 et 172.)

[Ferber, 1997] Ferber, J. (1997). Les systèmes multi-agents : un aperçu général. *Techniques et sciences informatiques*, 16(8). (Cité en pages 25 et 26.)

[Figer, 2005] Figer, J.-P. (2005). *Architectures orientées services SOA*. Ed. Techniques Ingénieur. (Cité en page 15.)

[Fki, 2015] Fki, E. (2015). *Sélection et composition flexible basée services abstraits pour une meilleure adaptation aux intentions des utilisateurs*. PhD thesis, Univerisité Toulouse 1 Capitole. (Cité en page 37.)

[Formo, 2012] Formo, J. (2012). A social web of things. *Ericsson User Experience Lab Blog*. (Cité en page 68.)

[Fürber and Hepp, 2010] Fürber, C. and Hepp, M. (2010). Using sparql and spin for data quality management on the semantic web. In *International Conference on Business Information Systems*, pages 35–46. Springer. (Cité en page 24.)

[Gandon, 2017] Gandon, F. (2017). Pour tout le monde : Tim berners-lee, lauréat du prix turing 2016 pour avoir inventé... le web. *1024 : Bulletin de la Société Informatique de France*, (11). (Cité en page 20.)

[Garton et al., 1997] Garton, L., Haythornthwaite, C., and Wellman, B. (1997). Studying online social networks. *Journal of computer-mediated communication*, 3(1) :JCMC313. (Cité en page 68.)

[Gilbert, 1997] Gilbert, D. (1997). Intelligent agents. *IBM Intelligent Agent Center*. (Cité en page 25.)

[Girau et al., 2016] Girau, R., Martis, S., and Atzori, L. (2016). Lysis : A platform for iot distributed applications over socially connected objects. *IEEE Internet of Things Journal*, 4(1) :40–51. (Cité en pages 74 et 75.)

[Girau et al., 2013] Girau, R., Nitti, M., and Atzori, L. (2013). Implementation of an experimental platform for the social internet of things. In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 500–505. IEEE. (Cité en pages 73 et 75.)

[Gortmaker et al., 2004] Gortmaker, J., Janssen, M., and Wagenaar, R. W. (2004). The advantages of web service orchestration in perspective. In *Proceedings of the*

6th international conference on Electronic commerce, pages 506–515. ACM. (Cité en page 38.)

[Grigori et al., 2010] Grigori, D., Corrales, J. C., Bouzeghoub, M., and Gater, A. (2010). Ranking bpel processes for service discovery. *IEEE Transactions on Services Computing*, 3(3) :178–192. (Cité en pages 32 et 42.)

[Grira et al., 2004] Grira, N., Crucianu, M., and Boujemaa, N. (2004). Unsupervised and semi-supervised clustering : a brief survey. *A review of machine learning techniques for processing multimedia content*, 1 :9–16. (Cité en page 81.)

[Gruber, 1991] Gruber, T. R. (1991). The role of common ontology in achieving sharable, reusable knowledge bases. *KR*, 91 :601–602. (Cité en page 21.)

[Guidara et al., 2015] Guidara, I., Guermouche, N., Chaari, T., Tazi, S., and Jmaiel, M. (2015). Heuristic based time-aware service selection approach. In *2015 IEEE International Conference on Web Services*, pages 65–72. IEEE. (Cité en page 36.)

[Guitton and Fiorino, 2006] Guitton, J. and Fiorino, H. (2006). Planification multi-agent pour la composition dynamique de services web. *Intelligence*. (Cité en page 38.)

[Han et al., 2013] Han, S. N., Lee, G. M., and Crespi, N. (2013). Semantic context-aware service composition for building automation system. *IEEE Transactions on industrial informatics*, 10(1) :752–761. (Cité en pages 48 et 52.)

[Hendler, 2009] Hendler, J. (2009). Web 3.0 emerging. *Computer*, 42(1) :111–113. (Cité en page 20.)

[Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al. (2004). Swrl : A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21(79) :1–31. (Cité en page 24.)

[Huget and Koning, 2001] Huget, M.-P. and Koning, J.-L. (2001). *Ingénierie des protocoles d'interaction pour les systèmes multi-agents.* PhD thesis, ANRT. (Cité en page 27.)

[Hussein et al., 2017] Hussein, D., Han, S. N., Lee, G. M., Crespi, N., and Bertin, E. (2017). Towards a dynamic discovery of smart services in the social internet of things. *Computers & Electrical Engineering*, 58 :429–443. (Cité en pages 72 et 75.)

[Hussein et al., 2015] Hussein, D., Park, S., Han, S. N., and Crespi, N. (2015). Dynamic social structure of things : A contextual approach in cpss. *IEEE Internet Computing*, 19(3) :12–20. (Cité en pages 72 et 75.)

[IBM, 2001] IBM (2001). Web services basics. available on http://www6.software.ibm.com/developerworks/education/wsbasics/wsbasics-ltr.pdf. (Cité en page 17.)

[IBM, 2014] IBM (2014). The structure of a soap message. available on https://www.ibm.com/support/knowledgecenter/en/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55780_.htm. (Cité en page 19.)

[Ibrahim and Ataelfadiel, 2018] Ibrahim, E. A. A. and Ataelfadiel, M. A. M. (2018). Ontology life cycle : A survey on the ontology and its development steps. (Cité en page 22.)

[iCore, 2018] iCore (2018). Projet icore. available on http://www.iot-icore.eu. (Cité en pages 50 et 52.)

[Iorga et al., 2017] Iorga, M., Feldman, L., Barton, R., Martin, M., Goren, N., and Mahmoudi, C. (2017). The nist definition of fog computing. Technical report, National Institute of Standards and Technology. (Cité en page 29.)

[ITU-T, 2012] ITU-T (2012). Présentation générale de l'internet des objets. ITU-T Study Group 20. (Cité en page 12.)

[Jalali et al., 2016] Jalali, F., Hinton, K., Ayre, R., Alpcan, T., and Tucker, R. S. (2016). Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5) :1728–1739. (Cité en page 30.)

[Jamont and Occello, 2006] Jamont, J.-P. and Occello, M. (2006). Une approche multi-agents pour la gestion de la communication dans les réseaux de capteurs sans fil. *Revue des Sciences et Technologies de l'Information-Série TSI : Technique et Science Informatiques*, 25(5) :661–690. (Cité en page 27.)

[Janowicz et al., 2019] Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., and Lefrançois, M. (2019). Sosa : A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56 :1–10. (Cité en page 22.)

[Jennings, 2010] Jennings, R. (2010). *Cloud computing with the Windows Azure platform*. John Wiley & Sons. (Cité en page 29.)

[Jiang et al., 2006] Jiang, H., Vidal, J. M., and Huhns, M. N. (2006). Incorporating emotions into automated negotiation. (Cité en page 27.)

[JoSEP et al., 2010] JoSEP, A. D., KAtz, R., KonWinSKi, A., Gunho, L., PAttERSon, D., and RABKin, A. (2010). A view of cloud computing. *Communications of the ACM*, 53(4). (Cité en page 29.)

[Juric, 2007] Juric, M. B. (2007). *SOA Approach to Integration : XML, Web services, ESB, and BPEL in real-world SOA projects*. Packt Publishing Ltd. (Cité en page 16.)

[Juric et al., 2006] Juric, M. B., Mathew, B., and Sarang, P. G. (2006). *Business process execution language for web services : an architect and developer's guide to orchestrating web services using BPEL4WS*. Packt Publishing Ltd. (Cité en page 92.)

[Kang and Sim, 2011] Kang, J. and Sim, K. M. (2011). Towards agents and ontology for cloud service discovery. In *2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pages 483–490. IEEE. (Cité en pages 33 et 42.)

[Karimi et al., 2017] Karimi, M. B., Isazadeh, A., and Rahmani, A. M. (2017). Qos-aware service composition in cloud computing using data mining techniques and genetic algorithm. *The Journal of Supercomputing*, 73(4) :1387–1415. (Cité en pages 36 et 42.)

[Kasnesis et al., 2017] Kasnesis, P., Patrikakis, C. Z., Kogias, D., Toumanidis, L., and Venieris, I. S. (2017). Cognitive friendship and goal management for the social iot. *Computers & Electrical Engineering*, 58 :412–428. (Cité en pages 73 et 75.)

[Kelaidonis et al., 2012] Kelaidonis, D., Somov, A., Foteinos, V., Poulios, G., Stavroulaki, V., Vlacheas, P., Demestichas, P., Baranov, A., Biswas, A. R., and Giaffreda, R. (2012). Virtualization and cognitive management of real world objects in the internet of things. In *2012 IEEE International Conference on Green Computing and Communications*, pages 187–194. IEEE. (Cité en pages 48, 52 et 53.)

[Khezami et al., 2005] Khezami, N., Otmane, S., and Mallem, M. (2005). Modelling and evaluation of a multi-agent system for a collaboration. *IFAC Proceedings Volumes*, 38(1) :559–564. (Cité en page 27.)

[Kibria et al., 2016] Kibria, M. G., Kim, H.-S., and Chong, I. (2016). Iot learning model based on virtual object cognition. In *2016 International Conference on Information Networking (ICOIN)*, pages 369–371. IEEE. (Cité en page 50.)

[Kim et al., 2016] Kim, M., Mohindra, A., Muthusamy, V., Ranchal, R., Salapura, V., Slominski, A., and Khalaf, R. (2016). Building scalable, secure, multi-tenant cloud services on ibm bluemix. *IBM Journal of Research and Development*, 60(2-3) :8–1. (Cité en page 29.)

[Klinpratum et al., 2014] Klinpratum, T., Saivichit, C., Elmangoush, A., and Magedanz, T. (2014). Toward interconnecting m2m/iot standards : Interworking proxy for ieee1888 standard at etsi m2m platform. In *29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC 2014), Phuket, Thailand*. (Cité en page 50.)

[Knublauch, 2017] Knublauch, H. (20 July 2017). Shapes constraint language (shacl) : W3c recommendation. Accessed : 2020-04-19. (Cité en page 24.)

[Knublauch et al., 2011] Knublauch, H., Hendler, J. A., and Idehen, K. (2011). Spin-overview and motivation. *W3C Member Submission*, 22 :W3C. (Cité en page 24.)

[Kolodner, 2014] Kolodner, J. (2014). *Case-based reasoning*. Morgan Kaufmann. (Cité en page 113.)

[Kopeckỳ et al., 2008] Kopeckỳ, J., Gomadam, K., and Vitvar, T. (2008). hrests : An html microformat for describing restful web services. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 619–625. IEEE. (Cité en page 23.)

[Kourtesis et al., 2008] Kourtesis, D., Paraskakis, and Iraklis (2008). Combining sawsdl, owl-dl and uddi for semantically enhanced web service discovery. In *European semantic web conference*, pages 614–628. Springer. (Cité en pages 33 et 42.)

[Kranz et al., 2010] Kranz, M., Roalter, L., and Michahelles, F. (2010). Things that twitter : social networks and the internet of things. In *What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010)*, pages 1–10. (Cité en pages 71 et 75.)

[Kuzminykh, 2018] Kuzminykh, I. (2018). Avatar conception for "thing" representation in internet of things. In *14th Swedish National Computer Networking Workshop, Karlskrona, Sweden*. (Cité en page 53.)

[Lassila et al., 1998] Lassila, O., Swick, R. R., et al. (1998). Resource description framework (rdf) model and syntax specification. (Cité en page 18.)

[Lejri and Tagina, 2012] Lejri, O. and Tagina, M. (2012). Representation in case-based reasoning applied to control reconfiguration. In *Industrial Conference on Data Mining*, pages 113–120. Springer. (Cité en page 114.)

[Liu and Liu, 2009] Liu, X. and Liu, Z. (2009). Research and implementation of eai based on soa and bpm. *Softw. Guide*, 8(3) :31–32. (Cité en pages xi et 16.)

[Liu et al., 2017] Liu, Y., Yang, R., and Zhang, S. (2017). Service selection method based on skyline in cloud environment. *International Journal of Performability Engineering*, 13(7) :1039–1047. (Cité en pages 35 et 42.)

[Liu and Xu, 2014] Liu, Z. and Xu, X. (2014). S-abc-a service-oriented artificial bee colony algorithm for global optimal services selection in concurrent requests environment. In *2014 IEEE International Conference on Web Services*, pages 503–509. IEEE. (Cité en page 35.)

[Lopez-Velasco, 2008] Lopez-Velasco, C. (2008). *Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation*. PhD thesis. (Cité en page 35.)

[Lu et al., 2012a] Lu, G., Seed, D., Starsinic, M., Wang, C., and Russell, P. (2012a). Enabling smart grid with etsi m2m standards. In *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 148–153. IEEE. (Cité en pages 50 et 52.)

[Lu et al., 2012b] Lu, G., Wang, T., Zhang, G., and Li, S. (2012b). Semantic web services discovery based on domain ontology. In *World Automation Congress 2012*, pages 1–4. IEEE. (Cité en page 33.)

[Maamar et al., 2011] Maamar, Z., Wives, L. K., Badr, Y., Elnaffar, S., Boukadi, K., and Faci, N. (2011). Linkedws : A novel web services discovery model based on the metaphor of "social networks". *Simulation Modelling Practice and Theory*, 19(1) :121–132. (Cité en pages 71 et 75.)

[MacQueen et al., 1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA. (Cité en page 81.)

[Maleshkova et al., 2009] Maleshkova, M., Pedrinaci, C., and Domingue, J. (2009). Supporting the creation of semantic restful service descriptions. (Cité en page 23.)

[Malik et al., 2019] Malik, S., Ahmad, S., and Kim, D. (2019). A novel approach of iot services orchestration based on multiple sensor and actuator platforms using virtual objects in online iot app-store. *Sustainability*, 11(20) :5859. (Cité en page 53.)

[Marengereke and Krishnan, 2015] Marengereke, T. and Krishnan, S. (2015). Cloud based security solution for android smartphones. pages 1–6. (Cité en pages 34 et 42.)

[Martin et al., 2004] Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., et al. (2004). Bringing semantics to web services : The owl-s approach. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 26–42. Springer. (Cité en page 19.)

[McDermott, 2002] McDermott, D. V. (2002). Estimated-regression planning for interactions with web services. In *AIPS*, volume 2, pages 204–211. (Cité en pages 39 et 42.)

[McGuinness and Da Silva, 2004] McGuinness, D. L. and Da Silva, P. P. (2004). Explaining answers from the semantic web : The inference web approach. *Web Semantics : Science, Services and Agents on the World Wide Web*, 1(4) :397–413. (Cité en page 23.)

[Médini et al., 2017] Médini, L., Mrissa, M., Khalfi, E.-M., Terdjimi, M., Le Sommer, N., Capdepuy, P., Jamont, J.-P., Occello, M., and Touseau, L. (2017). Building a web of things with avatars : A comprehensive approach for concern management in wot applications. In *Managing the Web of Things*, pages 151–180. Elsevier. (Cité en page 53.)

[Medjahed, 2004] Medjahed, B. (2004). *Semantic web enabled composition of web services*. PhD thesis, Virginia Tech. (Cité en pages 39 et 42.)

[Mei et al., 2011] Mei, A., Morabito, G., Santi, P., and Stefa, J. (2011). Social-aware stateless forwarding in pocket switched networks. In *2011 Proceedings IEEE INFOCOM*, pages 251–255. IEEE. (Cité en page 78.)

[Mell and Grance, 2011] Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Technical Report 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD. (Cité en page 29.)

[Mennis, 2006] Mennis, E. A. (2006). The wisdom of crowds : Why the many are smarter than the few and how collective wisdom shapes business, economies, societies, and nations. *Business Economics*, 41(4) :63–65. (Cité en page 2.)

[Mo et al., 2018] Mo, C., Li, Y., and Zheng, L. (2018). Simulation and analysis on overtaking safety assistance system based on vehicle-to-vehicle communication. *Automotive Innovation*, 1(2) :158–166. (Cité en page 61.)

[Mohr et al., 2015] Mohr, F., Jungmann, A., and Büning, H. K. (2015). Automated online service composition. In *2015 IEEE International Conference on Services Computing*, pages 57–64. IEEE. (Cité en pages 40 et 42.)

[Mokhtar et al., 2005] Mokhtar, S. B., Fournier, D., Georgantas, N., and Issarny, V. (2005). Context-aware service composition in pervasive computing environments. In *International Workshop on Rapid Integration of Software Engineering Techniques*, pages 129–144. Springer. (Cité en pages 39 et 42.)

[Motik et al., 2005] Motik, B., Sattler, U., and Studer, R. (2005). Query answering for owl-dl with rules. *Web Semantics : Science, Services and Agents on the World Wide Web*, 3(1) :41–60. (Cité en page 24.)

[Mrissa et al., 2015] Mrissa, M., Médini, L., Jamont, J.-P., Le Sommer, N., and Laplace, J. (2015). An avatar architecture for the web of things. *IEEE Internet Computing*, 19(2) :30–38. (Cité en pages 49 et 52.)

[Muller et al., 2006] Muller, I., Kowalczyk, R., and Braun, P. (2006). Towards agent-based coalition formation for service composition. In *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 73–80. (Cité en pages 34 et 42.)

[Müller, 2002] Müller, J. P. (2002). *Des systèmes autonomes aux systèmes multi-agents : Interaction, émergence et systèmes complexes.* PhD thesis, UM2. (Cité en page 28.)

[NeOn, 2015] NeOn (2015). The neon methodology. available on http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/methodologies/59-neon-methodology/. (Cité en page 56.)

[Nitti et al., 2014] Nitti, M., Atzori, L., and Cvijikj, I. P. (2014). Friendship selection in the social internet of things : challenges and possible strategies. *IEEE Internet of things journal*, 2(3) :240–247. (Cité en pages 71 et 75.)

[Nitti et al., 2015] Nitti, M., Pilloni, V., Colistra, G., and Atzori, L. (2015). The virtual object as a major element of the internet of things : a survey. *IEEE Communications Surveys & Tutorials*, 18(2) :1228–1240. (Cité en page 49.)

[ornikar, 2020] ornikar (2020). Les règles du croisement et du dépassement. disponible sur : https://www.ornikar.com/code/cours/circulation/croisements-depassements. (Cité en page 6.)

[Paik et al., 2017] Paik, H.-y., Lemos, A. L., Barukh, M. C., Benatallah, B., and Natarajan, A. (2017). Web services–rest or restful services. In *Web Service Implementation and Composition Techniques*, pages 67–91. Springer. (Cité en pages 19 et 20.)

[Palathingal et al., 2004] Palathingal, P., Chandra, and Sandeep (2004). Agent approach for service discovery and utilization. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 9–pp. IEEE. (Cité en pages 33 et 42.)

[Parodi et al., 2015] Parodi, A., Maresca, M., Provera, M., and Baglietto, P. (2015). An iot approach for the connected vehicle. In *International Internet of Things Summit*, pages 158–161. Springer. (Cité en page 50.)

[Pedrinaci et al., 2010] Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., and Domingue, J. (2010). iserve : a linked services publishing platform. In *CEUR workshop proceedings*, volume 596. (Cité en page 22.)

[Peer, 2004] Peer, J. (2004). A pddl based tool for automatic web service composition. In *International Workshop on Principles and Practice of Semantic Web Reasoning*, pages 149–163. Springer. (Cité en page 38.)

[Pintus et al., 2012] Pintus, A., Carboni, D., and Piras, A. (2012). Paraimpu : a platform for a social web of things. In *Proceedings of the 21st International Conference on World Wide Web*, pages 401–404. (Cité en pages 74 et 75.)

[Pradel et al., 2014] Pradel, C., Haemmerlé, O., and Hernandez, N. (2014). Swip : a natural language to sparql interface implemented with sparql. In *International Conference on Conceptual Structures*, pages 260–274. Springer. (Cité en page 157.)

[Prasad and Rao, 1990] Prasad, N. N. and Rao, J. N. (1990). The estimation of the mean squared error of small-area estimators. *Journal of the American statistical association*, 85(409) :163–171. (Cité en page 173.)

[Presser et al., 2009] Presser, M., Barnaghi, P. M., Eurich, M., and Villalonga, C. (2009). The sensei project : Integrating the physical world with the digital world of the network of the future. *IEEE Communications Magazine*, 47(4) :1–4. (Cité en page 49.)

[Presutti and Gangemi, 2008] Presutti, V. and Gangemi, A. (2008). Content ontology design patterns as practical building blocks for web ontologies. In *International Conference on Conceptual Modeling*, pages 128–141. Springer. (Cité en page 23.)

[Pukkasenung et al., 2010] Pukkasenung, P., Sophatsathit, P., and Lursinsap, C. (2010). An efficient semantic web service discovery using hybrid matching. In *Proceedings of the 2nd International Conference on Knowledge and Smart Technologies (KST2010)*, page 49. (Cité en page 32.)

[Qi et al., 2010] Qi, L., Tang, Y., Dou, W., and Chen, J. (2010). Combining local optimization and enumeration for qos-aware web service composition. In *2010 IEEE International Conference on Web Services*, pages 34–41. IEEE. (Cité en page 36.)

[Qiqing et al., 2009] Qiqing, F., Xiaoming, P., Qinghua, L., and Yahui, H. (2009). A global qos optimizing web services selection algorithm based on moaco for dynamic web service composition. In *2009 International forum on information technology and applications*, volume 1, pages 37–42. IEEE. (Cité en page 35.)

[Quinqueton, 2013] Quinqueton, J. (2013). Organisation des systèmes multi-agents. (Cité en page 26.)

[Rajeswari et al., 2014] Rajeswari, M., Sambasivam, G., Balaji, N., Basha, M. S., Vengattaraman, T., and Dhavachelvan, P. (2014). Appraisal and analysis on various web service composition approaches based on qos factors. *Journal of King Saud University-Computer and Information Sciences*, 26(1) :143–152. (Cité en pages 36 et 42.)

[Roman et al., 2005] Roman, D., Keller, U., Lausen, H., De Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. *Applied ontology*, 1(1) :77–106. (Cité en page 23.)

[Römer et al., 2004] Römer, K., Schoch, T., Mattern, F., and Dübendorfer, T. (2004). Smart identification frameworks for ubiquitous computing applications. *Wireless Networks*, 10(6) :689–700. (Cité en pages 48 et 52.)

[Safar and Reynaud, 2009] Safar, B. and Reynaud, C. (2009). Alignement d'ontologies basé sur des ressources complémentaires illustration sur le système taxomap. *Technique et Science Informatiques*, 28(10) :1211–1232. (Cité en page 21.)

[Sajadi et al., 2016] Sajadi, Z., Rastegari, Y., and Shams, F. (2016). Web service choreography verification using z formal specification. *International Journal of Engineering*, 29(11) :1549–1557. (Cité en page 38.)

[Sánchez A, 2003] Sánchez A, V. D. (2003). Advanced support vector machines and kernel methods. *Neurocomputing*, 55(1-2) :5–20. (Cité en page 118.)

[Sarang, 2007] Sarang, P. (2007). *SOA Approach to Integration : XML, Web Services, ESB, and BPEL in Real-World SOA Projects*. Packt Publishing. (Cité en page 16.)

[Seghir, 2018] Seghir, F. (2018). *Composition automatique de services web sémantiques par planification et techniques d'intelligence artificielle dans des systèmes ouverts et dynamiques*. PhD thesis. (Cité en page 38.)

[Sellami et al., 2016] Sellami, M., Bouzid, M., and Belleili, H. (2016). Formation de coalition pour la maximisation de l'utilité de la réponse : Application à la recherche d'information. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 3. (Cité en pages 34 et 42.)

[SENSEI, 2017] SENSEI (2017). Zigbee alliance. available on https://www.ict-sensei.org/. (Cité en pages 49 et 52.)

[Seydoux, 2018] Seydoux, N. (2018). *Towards interoperable IOT systems with a constraint-aware semantic web of things*. PhD thesis, INSA Toulouse. (Cité en page 3.)

[Seydoux et al., 2016] Seydoux, N., Drira, K., Hernandez, N., and Monteil, T. (2016). Iot-o, a core-domain iot ontology to represent connected devices networks. In *European Knowledge Acquisition Workshop*, pages 561–576. Springer. (Cité en pages 22 et 51.)

[Shamszaman et al., 2014] Shamszaman, Z., Ara, S., Chong, I., and Jeong, Y. (2014). Web-of-objects (woo)-based context aware emergency fire management systems for the internet of things. *Sensors*, 14(2) :2944–2966. (Cité en pages 48 et 52.)

[Shamszaman and Ali, 2017] Shamszaman, Z. U. and Ali, M. I. (2017). Toward a smart society through semantic virtual-object enabled real-time management framework in the social internet of things. *IEEE Internet of Things Journal*, 5(4) :2572–2579. (Cité en page 53.)

[Sheshagiri et al., 2003] Sheshagiri, M., Finin, T., et al. (2003). A planner for composing services described in daml-s. In *Proceedings of the AAMAS Workshop on Web Services and Agent-based Engineering,*. (Cité en page 38.)

[Shoham, 1990] Shoham, Y. (1990). Agent–oriented programming, rapport technique no. Technical report, STAN–CS–90–1335, Computer Science Department, Standford University. (Cité en page 25.)

[Sirin and Parsia, 2004] Sirin, E. and Parsia, B. (2004). Planning for semantic web services. In *SWS@ ISWC*. (Cité en page 39.)

[Smets et al., 2007] Smets, K., Verdonk, B., and Jordaan, E. (2007). Evaluation of performance measures for svr hyperparameter selection. pages 637 – 642. (Cité en page 145.)

[Srinivasan et al., 2004] Srinivasan, N., Paolucci, M., and Sycara, K. (2004). Adding owl-s to uddi, implementation and throughput. *Proceedings of Semantic Web Service and Web Process Composition*, 25. (Cité en pages 33 et 42.)

[Stahl, 2003] Stahl, A. (2003). *Learning of knowledge-intensive similarity measures in case-based reasoning*. dissertation. de. (Cité en page 114.)

[Stypsanelli et al., 2020] Stypsanelli, I.-V., Medjiah, S., and Prabhu, B. J. (2020). Reducing service migrations in fog infrastructures by optimizing node location. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 13–19. IEEE. (Cité en page 65.)

[Sycara et al., 2004] Sycara, K., Paolucci, M., Soudry, J., and Srinivasan, N. (2004). Dynamic discovery and coordination of agent-based semantic web services. *IEEE Internet computing*, 8(3) :66–73. (Cité en page 33.)

[thingspeak, 2019] thingspeak (2019). thingspeak. (Cité en pages 73 et 75.)

[Tietz et al., 2011] Tietz, V., Blichmann, G., Pietschmann, S., and Meißner, K. (2011). Task-based recommendation of mashup components. In *International Conference on Web Engineering*, pages 25–36. Springer. (Cité en page 59.)

[Tim, 1994] Tim, B. L. (1994). Plenary talk by tim bl at wwwf94 : Overview. Accessed : 2020-03-21. (Cité en page 20.)

[Tran and Wagner, 2000] Tran, D. and Wagner, M. (2000). Fuzzy entropy clustering. In *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*, volume 1, pages 152–157. IEEE. (Cité en page 172.)

[Travers and Milgram, 1967] Travers, J. and Milgram, S. (1967). The small world problem. *Phychology Today*, 1(1) :61–67. (Cité en page 69.)

[Travers and Milgram, 2006] Travers, J. and Milgram, S. (2006). An experimental study of the small world problem. *The structure and dynamics of networks*, pages 130–148. (Cité en page 85.)

[Traverso and Pistore, 2004] Traverso, P. and Pistore, M. (2004). Automated composition of semantic web services into executable processes. In *International Semantic Web Conference*, pages 380–394. Springer. (Cité en page 39.)

[Vaquero and Rodero-Merino, 2014] Vaquero, L. M. and Rodero-Merino, L. (2014). Finding your way in the fog : Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5) :27–32. (Cité en page 30.)

[Vautherot, 2007] Vautherot (2007). La création d'internet. disponible sur : http://www.gralon.net/articles/internet-et-webmaster/logiciel/article-la-creation-d-internet-130.htm. (Cité en page 1.)

[Vlacheas et al., 2013] Vlacheas, P., Giaffreda, R., Stavroulaki, V., Kelaidonis, D., Foteinos, V., Poulios, G., Demestichas, P., Somov, A., Biswas, A. R., and Moessner, K. (2013). Enabling smart cities through a cognitive management framework for the internet of things. *IEEE communications magazine*, 51(6) :102–111. (Cité en page 53.)

[W3C, 2004] W3C (2004). Web services architecture. available on https://www.w3.org/TR/ws-arch/#whatis. (Cité en page 17.)

[W3C, 2015] W3C (2015). Introduction to the web of things. available on https://www.w3.org/2015/03/intro-wot.pdf. (Cité en page 53.)

[W3C, 2016] W3C (2016). Architecture de web of things (wot). disponible sur : https://www.w3.org/TR/2017/WD-wot-architecture-20170914/. (Cité en page 13.)

[Wagner and Trautmann, 2010] Wagner, T. and Trautmann, H. (2010). Integration of preferences in hypervolume-based multiobjective evolutionary algorithms by means of desirability functions. *IEEE Transactions on Evolutionary Computation*, 14(5) :688–701. (Cité en page 172.)

[Wang et al., 2015] Wang, G., Ju, Y., Carr, T. R., and Tan, F. (2015). The hierarchical decomposition method and its application in recognizing marcellus shale lithofacies through combining with neural network. *Journal of Petroleum Science and Engineering*, 127 :469 – 481. (Cité en page 145.)

[Wang et al., 2011] Wang, S.-G., Sun, Q.-B., and Yang, F.-C. (2011). Web service dynamic selection by the decomposition of global qos constraints. *Ruanjian Xuebao/Journal of Software*, 22(7) :1426–1439. (Cité en pages 92 et 108.)

[Wang et al., 2010] Wang, W., Sun, Q., Zhao, X., and Yang, F. (2010). An improved particle swarm optimization algorithm for qos-aware web service selection in service oriented communication. *International Journal of Computational Intelligence Systems*, 3(sup01) :18–30. (Cité en pages 36 et 42.)

[Wang and Zhang, 2007] Wang, W. and Zhang, Y. (2007). On fuzzy cluster validity indices. *Fuzzy sets and systems*, 158(19) :2095–2117. (Cité en pages 171 et 172.)

[Watson, 1999] Watson, I. (1999). Case-based reasoning is a methodology not a technology. In *Research and Development in Expert Systems XV*, pages 213–223. Springer. (Cité en page 113.)

[Webofthings, 2016] Webofthings (2016). Architecting the web of things, for techies and thinkers ! disponible sur : https://webofthings.org/. (Cité en page 13.)

[Weerawarana et al., 2005] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F. (2005). *Web services platform architecture : SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR. (Cité en page 18.)

[Wen et al., 2017] Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J., and Rovatsos, M. (2017). Fog orchestration for internet of things services. *IEEE Internet Computing*, 21(2) :16–24. (Cité en page 38.)

[White et al., 2018] White, G., Palade, A., Cabrera, C., and Clarke, S. (2018). Iotpredict : Collaborative qos prediction in iot. In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–10. (Cité en page 135.)

[Wu et al., 2003] Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D. (2003). Automating daml-s web services composition using shop2. In *International Semantic Web Conference*, pages 195–210. Springer. (Cité en pages 39 et 42.)

[Xie and Beni, 1991] Xie, X. L. and Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on pattern analysis and machine intelligence*, 13(8) :841–847. (Cité en page 171.)

[Xu, 2015] Xu, W. (2015). *Modeling and exploiting the knowledge base of web of things*. PhD thesis. (Cité en page 13.)

[Yan and Zhi-Zhong, 2017] Yan, H. and Zhi-Zhong, L. (2017). Research on dynamic prediction method of qos of cloud service. *Journal of Software Engineering*, 11 :1–11. (Cité en page 120.)

[Yannuzzi et al., 2014] Yannuzzi, M., Milito, R., Serral-Gracià, R., Montero, D., and Nemirovsky, M. (2014). Key ingredients in an iot recipe : Fog computing, cloud computing, and more fog computing. In *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 325–329. IEEE. (Cité en page 30.)

[Yanwei et al., 2010] Yanwei, Z., Hong, N., Haojiang, D., and Lei, L. (2010). A dynamic web services selection based on decomposition of global qos constraints. In *2010 IEEE Youth Conference on Information, Computing and Telecommunications*, pages 77–80. IEEE. (Cité en pages 37 et 42.)

[Yi et al., 2015a] Yi, S., Hao, Z., Qin, Z., and Li, Q. (2015a). Fog computing : Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78. IEEE. (Cité en page 30.)

[Yi et al., 2015b] Yi, S., Li, C., and Li, Q. (2015b). A survey of fog computing : concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42. ACM. (Cité en page 29.)

[Younes et al., 2018] Younes, W., Trouilhet, S., Adreit, F., and Arcangeli, J.-P. (2018). Towards an intelligent user-oriented middleware for opportunistic composition of services in ambient spaces. In *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things*, pages 25–30. (Cité en pages 34 et 42.)

[Yu et al., 2007] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1) :6–es. (Cité en pages 36 et 42.)

[Yuan et al., 2019] Yuan, Y., Zhang, W., Zhang, X., and Zhai, H. (2019). Dynamic service selection based on adaptive global qos constraints decomposition. *Symmetry*, 11(3) :403. (Cité en pages 37 et 42.)

[Zaheer and al., 2012] Zaheer, K. and al. (2012). Future internet : The internet of things architecture, possible applications and key challenges. In *2012 10th International Conference on Frontiers of Information Technology*, pages 257–260. (Cité en pages xi et 14.)

[Zeng et al., 2008] Zeng, L., Ngu, A. H., Benatallah, B., Podorozhny, R., and Lei, H. (2008). Dynamic composition and optimization of web services. *Distributed and Parallel Databases*, 24(1-3) :45–72. (Cité en pages 40 et 42.)

[Zhang et al., 2010] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing : state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1) :7–18. (Cité en page 29.)

[Zheng et al., 2014] Zheng, Z., Zhang, Y., and Lyu, M. R. (2014). Investigating qos of real-world web services. *IEEE Transactions on Services Computing*, 7(1) :32–39. (Cité en page 135.)

[Zhou et al., 2009] Zhou, B., Huang, T., Liu, J., and Shen, M. (2009). Using inverted indexing to semantic web service discovery search model. In *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE. (Cité en pages 32 et 42.)

[Zhou et al., 2014] Zhou, K., Fu, C., and Yang, S. (2014). Fuzziness parameter selection in fuzzy c-means : The perspective of cluster validation. *Science China Information Sciences*, 57(11) :1–8. (Cité en page 131.)

[Zhovtobryukh, 2007] Zhovtobryukh, D. (2007). A petri net-based approach for automated goal-driven web service composition. *Simulation*, 83(1) :33–63. (Cité en page 157.)

Résumé : L'Internet des Objets (IoT) est une tendance très en vogue actuellement qui permet d'interconnecter un nombre important d'objets hétérogènes issus de plusieurs domaines différents. Le nombre de ces objets connectés ne cesse d'exploser à tel point que notre société sera submergée par des milliards d'objets dans un futur proche. Notre thèse vise principalement à proposer un modèle pour la modélisation des objets IoT hétérogènes sur le Web via des avatars qui sont également dotés de capacités de raisonnement et de décision. Ensuite, une approche de découverte est proposée pour permettre à un avatar donné de localiser les services IoT requis pour réaliser ses objectifs complexes dans un système à large échelle. Elle est basée sur les mécanismes de réseautage social et les algorithmes de clustering. Après cela, une approche de sélection basée sur la QoS est proposée pour choisir les services les plus adéquats parmi les services IoT découverts. Elle est réalisée via : une méthode basée sur un algorithme génétique évolutionnaire multi-objectifs (MOEA) pour la décomposition des contraintes de QoS globales et une méthode basée sur la régression par machine à vecteurs de support (SVM). Les approches proposées sont évaluées et expliquées en moyennant un scénario de dépassement des véhicules connectés.
Mots clés : IoT, Avatar, Sémantique, Réseaux sociaux, Clustering, QoS, MOEA, SVM, Véhicules connectés.

Abstract : The Internet of Things (IoT) is a very popular trend today which allows a large number of heterogeneous objects from several different fields to be interconnected. The number of these connected objects continues to explode to such an extent that our society will be submerged by billions of objects in the near future. Our thesis mainly aims to propose a model for the modeling of heterogeneous IoT objects on the Web via avatars which are also endowed with reasoning and decision capacities. Next, a discovery approach is proposed to enable a given avatar to locate the IoT services required to achieve its complex goals in a large scale system. It is based on social networking mechanisms and clustering algorithms. After that, a QoS-based selection approach is proposed to choose the most suitable services from the discovered IoT services. It is carried out via : a method based on a multi-objective evolutionary genetic algorithm (MOEA) for decomposition of global QoS constraints and a method based on support vector machine regression (SVM). The proposed approaches are evaluated and explained by averaging a scenario of overtaking connected vehicles.

Keywords : IoT, avatar, semantic, social networks, clustering, QoS, MOEA, SVM, connected vehicles.